Thesis for The Degree of Doctor of Philosophy

On the Secure and Resilient Design of Connected Vehicles: Methods and Guidelines

Thomas Rosenstatter

CHALMERS
UNIVERSITY OF TECHNOLOGY

Division of Computer and Network Systems
Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2021

On the Secure and Resilient Design of Connected Vehicles: Methods and Guidelines

Thomas Rosenstatter

ISBN 978-91-7905-533-2
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 5000.
ISSN 0346-718X

Technical Report No 206D
Department of Computer Science & Engineering
Division of Computer and Network Systems
Chalmers University of Technology
Gothenburg, Sweden

Author e-mail: thomas.rosenstatter@chalmers.se,
rosenstatter.thomas@gmail.com

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2021.

To my father Franz.

On the Secure and Resilient Design of Connected Vehicles: Methods and Guidelines

Thomas Rosenstatter
Department of Computer Science and Engineering,
Chalmers University of Technology

Abstract

Vehicles have come a long way from being purely mechanical systems to systems that consist of an internal network of more than 100 microcontrollers and systems that communicate with external entities, such as other vehicles, road infrastructure, the manufacturer's cloud and external applications. This combination of resource constraints, safety-criticality, large attack surface and the fact that millions of people own and use them each day, makes securing vehicles particularly challenging as security practices and methods need to be tailored to meet these requirements.

This thesis investigates how security demands should be structured to ease discussions and collaboration between the involved parties and how requirements engineering can be accelerated by introducing generic security requirements. Practitioners are also assisted in choosing appropriate techniques for securing vehicles by identifying and categorising security and resilience techniques suitable for automotive systems. Furthermore, three specific mechanisms for securing automotive systems and providing resilience are designed and evaluated.

The first part focuses on cyber security requirements and the identification of suitable techniques based on three different approaches, namely (i) providing a mapping to security levels based on a review of existing security standards and recommendations; (ii) proposing a taxonomy for resilience techniques based on a literature review; and (iii) combining security and resilience techniques to protect automotive assets that have been subject to attacks.

The second part presents the design and evaluation of three techniques. First, an extension for an existing freshness mechanism to protect the in-vehicle communication against replay attacks is presented and evaluated. Second, a trust model for Vehicle-to-Vehicle communication is developed with respect to cyber resilience to allow a vehicle to include trust in neighbouring vehicles in its decision-making processes. Third, a framework is presented that enables vehicle manufacturers to protect their fleet by detecting anomalies and security attacks using vehicle trust and the available data in the cloud.

Keywords: security, resilience, cyber-physical systems, automotive, V2X, in-vehicle network, secure communication

Acknowledgments

First and foremost, I would like to thank my supervisor Tomas Olovsson and co-supervisor Magnus Almgren for their support, guidance and encouragement throughout my PhD studies. I also extend my thanks to Cristofer Englund for motivating me to pursue a PhD.

I am also grateful to my co-authors and collaborators; thank you for the fruitful discussions and feedback. A special thanks also goes to Volvo Trucks for having me in their office on Fridays during the course of the HoliSec project. Thank you, Christian Sandberg and Mahboobeh Daftari for the discussions and updates on interesting challenges.

I am honoured to have Prof. Eric Sax as the faculty opponent during the defence. I would also like to thank the members of the grading committee: Prof. Tuomas Aura, Prof. Simone Fischer-Hübner, Assoc. Prof. Robert Lagerström and Prof. Andrei Sabelfeld for reviewing my work and participating in my defence. Many thanks to my examiner Per Larsson-Edefors and the follow-up groups for supporting me during my studies.

Next, I'd like to thank my colleagues at the department who made Chalmers a great place to work. Thank you for the many fikas and after-work get-togethers. To Georgia, Babis, Dimitrios and Christos who (accidentally) started teaching me some, yet the most interesting phrases in Greek I need to know. To all past and present colleagues: Ali, Aljoscha, Aras, Bastian, Beshr, Boel, Carlo, Fazeleh, Francisco, Hannah, Iulia, Ivan, Karl, Kim, Marina, Nasser, Philippas, Valentin, Vincenzo and Wissam. A special thanks to the department's administration for their support during the past five years: Eva Axelsson, Marianne Pleen-Schreiber, Clara Oders, Patrik Johansson, Lars Norèn and Michael Morin.

To all my friends near and far, thank you for being with me all this time. It doesn't matter how often we manage to meet; it always feels like we last met yesterday. Every one of you knows how much you mean to me, and I truly don't know what I would do without you.

Lastly, I want to express my deepest gratitude to my family, without you nothing would have been possible. I left for an exchange year and now it has been six years that I live abroad. Much has happened during this time; we shared great moments together and I became Gödi of my lovely niece Jana, but we also faced a massive loss. I cannot express how grateful I am for having you and your support all this time.

This research was partially supported by the Swedish VINNOVA FFI Projects "HoliSec: Holistic approach to improve data security of vehicles" and "CyReV: Cyber Resilience for Vehicles – Cybersecurity for Automotive Systems in a Changing Environment (phase 1 and 2)" and the Swedish CoAct project.

Thomas Rosenstatter
Göteborg, August 2021

List of Publications

This thesis is based on the following appended publications:

Generic Security Requirements and Identification of Suitable Techniques

Paper A **T. Rosenstatter**, T. Olovsson, "Open Problems when Mapping Automotive Security Levels to System Requirements," in *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems - Volume 1: VEHITS*, pp. 451-260, ScitePress 2018.

Paper B **T. Rosenstatter**, T. Olovsson, "Towards a Standardized Mapping from Automotive Security Levels to Security Mechanisms," in *21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1501–1507, IEEE 2018.

Paper C **T. Rosenstatter**, K. Strandberg, R. Jolak, R. Scandariato, T. Olovsson, "REMIND: A Framework for the Resilient Design of Automotive Systems,"in *2020 IEEE Secure Development (SecDev)*, pp. 81–95, IEEE 2020.

Paper D K. Strandberg, **T. Rosenstatter**, R. Jolak, N. Nowdehi, T. Olovsson, "Resilient Shield: Reinforcing the Resilience of Vehicles Against Security Threats," in *2021 IEEE 93th Vehicular Technology Conference (VTC2021-Spring)*, pp. 1–7, IEEE 2021.

Design and Evaluation of Security and Resilience Techniques

Paper E **T. Rosenstatter**, C. Sandberg, T. Olovsson, "Extending AUTOSAR's Counter-based Solution for Freshness of Authenticated Messages in Vehicles," in *IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 1–10, IEEE 2019.

Paper F M. Aramrattana, J. Detournay, C. Englund, V. Fridmodig, O. Uddman Jansson, T. Larsson, W. Mostowski, V. Díez Rodríguez, **T. Rosenstatter**, G. Shahanoor, "Team Halmstad Approach to Cooperative Driving in the Grand Cooperative Driving Challenge 2016," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1248–1261, April 2018.

Paper G **T. Rosenstatter**, C. Englund, "Modelling the Level of Trust in a Cooperative Automated Vehicle Control System," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1237–1247, April 2018.

Paper H **T. Rosenstatter**, T. Olovsson, M. Almgren, "V2C: A Trust-Based Vehicle to Cloud Anomaly Detection Framework for Automotive Systems", *16th International Conference on Availability, Reliability and Security (ARES)*, pp. 1-10, ACM 2021.

Contents

Part I

Thesis Overview

Introduction

Vehicles have progressed from being purely mechanical, e. g., steam-powered, to include electromechanical components, such as pumps, relays and sensors, and to finally become mechatronic systems that make use of software to improve and create new functionality. In the last decade vehicles, especially cars and heavy-duty vehicles such as trucks, have started to evolve from being mechatronic systems to being enhanced with functionality using software inside and outside the vehicle. In other words, these automotive systems are evolving to a Cyber-Physical System (CPS) [1] where a single vehicle is seen only as a part of the system, and new functionality is achieved by software communicating to several entities such as other vehicles, road infrastructure, the OEM and other service providers.

The technological progress in computation and communication paved the way for a manifold of new concepts and trends that already influenced and accelerated the change of vehicles to become *computers on wheels*. The first Internet-connected applications can already be found in consumer vehicles, for example, services such as automated emergency calls (eCall), remote unlock of the vehicle, map synchronisation between the mobile app and the vehicle as well as remote updates of software inside the vehicle.

But the Internet is not the only passage to communicate with other entities. Research and industry introduced the concept of *Cooperative Intelligent Transport Systems (C-ITS)* [2] where vehicles use Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication to directly communicate with each other in order to increase traffic safety and efficiency [3]. This direct communication enables the vehicles to sense out-of-sight objects, e. g., receive warnings about road conditions and information about approaching vehicles, and directly cooperate with other vehicles when, for example, crossing an intersection.

Moving towards automated and even autonomous driving increases the functional requirements on vehicles to withstand various traffic situations in which vehicles may not be able to completely aware of the situation. In addition, hardware or software failures may occur and vehicles may be attacked by an individual or organisation. The challenge when coping with all these situations is that autonomous vehicles need to make all decisions autonomously without requiring manual intervention by their passengers as there is no driver with enough competence or operator left to intervene.

Contrary to this fast development of new functionality and increased connectivity in the last two decades caused by the rapid advances in communication and computation, physical security has long been the centre of security concerns. Discussions on the necessity of cyber security measures in vehicles accelerated within the research community thanks to the research done by Koscher et al. [4] and Checkoway et al. [5] in 2010 respectively 2011.

Outline. This thesis is structured as follows. Part I provides an overview of this thesis and gives an introduction to security and resilience in automotive systems. Section 1 describes the motivation for this thesis and also highlights the characteristics of the different types of communication and applications found in the domain. Section 2 presents high-level introductions to relevant areas to which this thesis contributes and Section 3 gives details of the research questions this thesis is addressing and connects them to the respective chapters. Section 4 provides details about the specific contributions of this thesis. The overall findings and contributions are then concluded in Section 5. Part II and Part III contain the explicit research contributions. Part II shows our work on defining security demands and ways to support practitioners in choosing appropriate cyber security and resilience techniques and Part III covers the design and evaluation of mechanisms that are proposed in this thesis.

1 Motivation

Functions and services for automotive systems can be divided into three types, namely (i) in-vehicle functions; (ii) cooperative driving functions; and (iii) cloud-based services. Figure 1 gives an overview of the characteristics of the components including examples of applications. Focusing on the differences shows that deploying security mechanisms is not trivial when considering, for instance, the characteristics and requirements of the communication in the in-vehicle network and the Vehicular Ad-hoc NETworks (VANETs).

Considering the aforementioned trends and developments that automotive systems have or are undergoing highlights that cyber security and resilience measures have become a necessity to protect drivers, passengers and everyone around this environment from individuals and organisations with malicious intent.

In-vehicle. The internal network of a single vehicle (in-vehicle network) builds the foundation for providing core functionalities, starting from being able to manually drive the vehicle and control exterior lighting to more automated driving functions such as Cruise Control (CC) and lane keeping, and eventually to reach autonomous driving functionality. Yet, the vehicle also needs to provide comfort functions to inform and entertain passengers, control interior lighting and offer connectivity.

The design and development of vehicles have long concentrated on building safe vehicles and therefore lacked common security practices. The importance and need for safe vehicles is also reflected in the early introduction of crash tests to evaluate vehicle safety and later with the release of the standard for functional safety of Electrical and/or Electronic (E/E) systems in road vehicles, ISO 26262 [6]. Hence, most technologies in vehicles were developed concentrating on safety and reliability

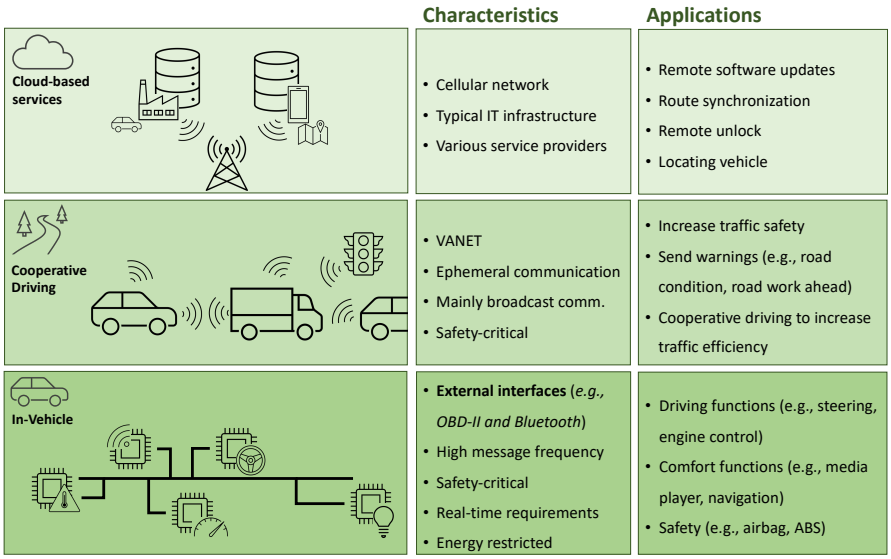


Figure 1: An overview of an automotive system, examples of involved components, typical characteristics and applications.

rather than security. A prominent example is the Controller Area Network (CAN) bus [7] developed in 1983 which is still used in safety-critical segments of the network in modern vehicles. CAN was not designed with security in mind and therefore lacks mechanisms to ensure confidentiality, integrity, availability, authenticity and freshness of the network traffic [8]. For this reason, the CAN bus has become a common target of cyber attacks [8–12].

Furthermore, modern vehicles consist of a complex internal network comprising of up to 150 resource-constrained microcontrollers, so-called Electronic Control Units (ECUs) [13]. Clearly, not all of these ECUs are powerful enough to perform cryptographic operations and still fulfil the real-time requirements needed in automotive systems, nor are they all critical for ensuring safety and security. However, the lack of additional computational resources, limited network bandwidth and the need for low energy consumption are reasons why well-established solutions for IT systems may not be usable or have to be adapted for these systems. It also needs to be noted that more computationally powerful ECUs also produce more heat which needs to be reflected in the hardware design. Besides, the costs for upgrading all these ECUs for security purposes have a direct impact on the OEM's revenue. For these reasons, it is essential to (i) include security assessment and design in the design process of vehicles and to not add security as an afterthought; and (ii) adapt techniques to include the key requirements of automotive systems, such as safety and timing constraints.

The lack of defined cyber security processes for Electrical and/or Electronic (E/E) systems in vehicles was first addressed by SAE J3061 [14], the *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*, which further led to the joint effort between

SAE and ISO to create a standard for cyber security engineering and management considering all stages of a vehicle's lifecycle. ISO/SAE 21434 [15] provides a common terminology for cyber security engineering and defines the requirements for cyber security risk assessment as well as requirements spanning all stages from engineering, production, operation, maintenance and decommissioning. At the time of writing this thesis ISO 21434 was under development, but has been published in August 2021.

External interfaces to the in-vehicle network. Providing communication to external entities outside of the vehicle is necessary for many reasons, such as to provide access for diagnostics, to upgrade firmware, and to enable mobile phones to connect to the infotainment system via Bluetooth. Researchers [5] have already shown that attackers do not need to perform physical modifications on the vehicle anymore, as the diagnostic port (OBD-II), Bluetooth interface and CD media player had vulnerabilities that made it possible to reprogram the firmware of an ECU and subsequently read and send arbitrary messages on the internal bus. Long distance communication via cellular networks has also shown to be insufficiently secured in the past [11].

Cooperative driving. V2V and V2I communication, in short Vehicle-to-Everything (V2X), are seen as an enabling technology for VANETs to increase traffic safety by having vehicles and road infrastructure directly exchange Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs). In Europe, this communication and use cases for cooperative driving are defined by the ETSI ITS committee.[1] This type of communication is challenging from a safety and security perspective as (i) vehicles within a VANET are highly mobile and may thus communicate only for a few seconds with each other; (ii) messages need to be cryptographically secured to ensure that the sending entities are authorised and messages are authentic; and (iii) vehicles may need to react fast upon receiving messages such as an emergency brake warning.

Today, attack scenarios [16, 17] are mainly theoretical or simulated due to the fact that VANET-enabled vehicles are not yet in serial production. There are, however, already test sites across Europe investigating V2X communication [18]. Furthermore, the information from authenticated messages received from legitimate vehicles may also be incorrect due to inaccurate or faulty sensors, but also due to vehicles misinterpreting the current situation. Therefore, it is important to validate the received information using trust or reputation models [19, 20] which include plausibility checks for information as well as additional evaluations for specific events (*data-oriented trust*) or ratings of a particular vehicle (*entity-oriented trust*) [21]. The need for plausibility checks in VANETs is also identified by ETSI ITS TR 102 893 [22].

Cloud-based services. Many comfort functions require support from the server backend of the OEM. Examples of such services are remote unlock, access to recent trip statistics and charging status via a smartphone or web application and map/trip synchronisation. Moreover, over-the-air updates and remote diagnostics are major improvements for both customers and OEMs because vehicle software can be updated remotely and more regularly without needing the customer to drive to an authorised

[1] `https://www.etsi.org/committee/its`

workshop and remote diagnostics can help to order spare parts or schedule repairs before being physically examined at the workshop.

Cloud services may also suffer from security vulnerabilities on the server-side, e.g., cross site scripting (XSS) and man-in-the-middle attacks [23–26]. From a security perspective, remote updates and diagnostics offer new opportunities to identify anomalies in vehicles and allow OEMs to quickly react to newly found vulnerabilities. For instance, BMW was able to patch a security vulnerability where 2.2 million vehicles were updated remotely over the air without the need for their customers to visit a workshop [23]. In addition to the advantages of fast software deployment, the data which is stored in the cloud and metadata of the communication can also be useful for security analysis to identify attacks.

Cyber resilience. With the increasing automation of vehicle driving functions, connectivity and consequently the potential effects caused by cyber attacks, a new quality next to reliability, safety and security is needed, namely *resilience*, the ability to endure and withstand cyber attacks. Unlike security techniques which typically focus on detecting and preventing cyber attacks, resilience adds the requirement to the system to maintain the intended operation, possibly with degraded functionality, when anomalies like attacks and faults occur [27].

Providing cyber security and resilience to vehicles requires measures for all types of communication within automotive systems. Only securing the vehicle itself, i. e., the communication within the in-vehicle network, is not sufficient anymore as new applications need connectivity to other vehicles and the cloud and thus increase the attack surface tremendously (as shown by Miller and Valasek [11]). Furthermore, utilising cryptographic solutions such as message authentication pose different challenges depending on the type of communication used within the automotive system. For example, cloud to vehicle communication via the Internet may be implemented using TLS with X.509 certificates, and in contrast, V2V communication additionally requires solutions for using temporary pseudonym certificates to offer privacy and mechanisms to revoke them promptly.

2 Background

This section presents a high-level introduction to the focus areas to which this thesis contributes, namely for *Part II*: methodologies to define cyber security requirements (Section 2.1); and the identification and categorisation of methods to guide practitioners in selecting suitable cyber security and resilience techniques (Section 2.2); for *Part III*: the protection of the in-vehicle communication with focus on the CAN bus (Section 2.3); and vehicle trust and anomaly detection in VANETs (Section 2.4). Figure 2 provides an overview of the thesis structure and refers to the relevant background sections for each chapter in this thesis.

2.1 Cyber Security Requirements

Cyber security requirements are based on the assessment of the system during the concept phase as defined in ISO 26262 and ISO 21434. In this section, a simplified

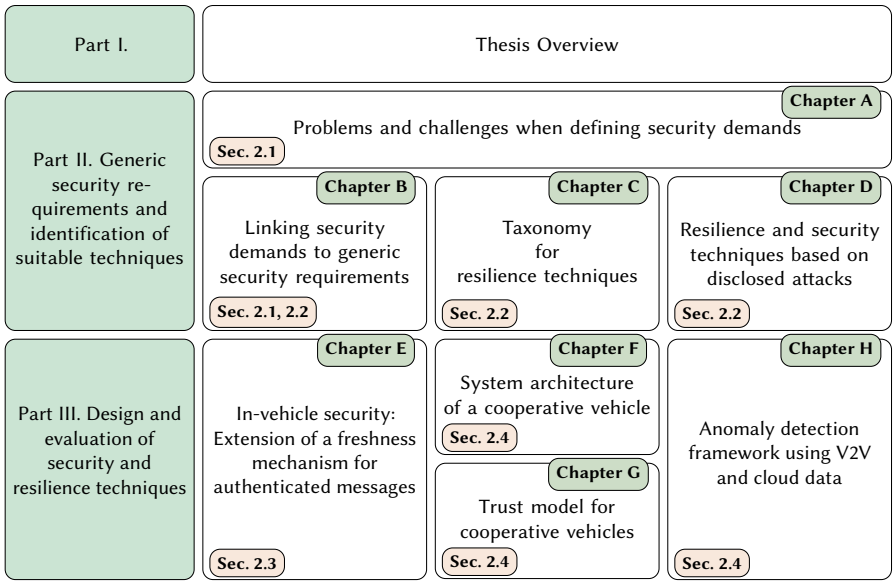


Figure 2: Overview of the structure of this thesis including references to the relevant background sections.

version of an in-vehicle network is described first, then automotive-specific methods for identifying and assessing security threats (TARA) are introduced including ways for how security requirements are derived and formulated in proposed methods. Thereafter, efforts to classify security and safety demands and methods to derive security requirements are presented. This thesis does not propose a new TARA technique, however, TARA needs to be considered when classifying security demands and further formulate a mapping to generic security requirements.

The in-vehicle network comprising 100 or more ECUs fulfils various driving and comfort functions. Figure 3 shows the reference architecture developed during the HoliSec project [28] showing a reduced version of an in-vehicle network of a modern vehicle with gateways interconnecting various subnets using different network technologies, such as CAN and Ethernet. To highlight the interaction between various components, the Cruise Control (CC) is briefly explained. The CC function requires the *Vehicle ECU* to send the target speed to the *Engine* and *Brake ECUs* to maintain the set speed, but also to send this information to the *Driver Display ECU* to provide the driver with visual feedback via the dashboard. Given this already simplified architecture of the in-vehicle network and possible use cases like the CC, it is evident that a methodology for a threat analysis and risk assessment is required to design and develop secure systems which consist of more than 100 components communicating with each other in order to fulfil automated driving functionality.

Threat Analysis and Risk Assessment (TARA). TARA is performed to identify and evaluate threats to a system and the associated risks. The outcome of a TARA is commonly used to derive security requirements that reduce the likelihood of

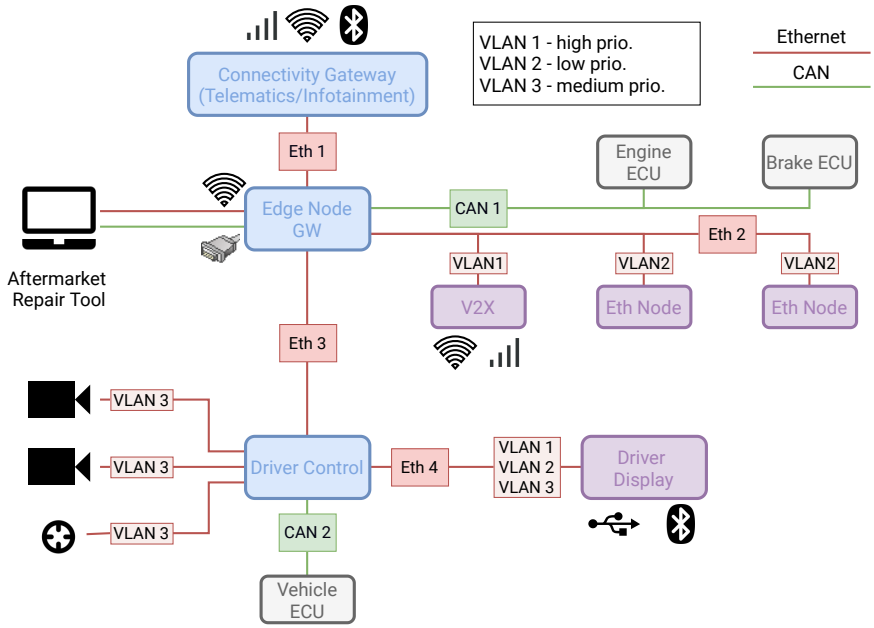


Figure 3: HoliSec reference architecture of a connected vehicle [29].

a threat or risk from happening. The automotive cyber security guidebook, i. e., SAE J3061 [14], splits TARA into three distinct phases; the threat analysis (*threat identification*), risk assessment (*threat classification*), and risk analysis (*threat ranking* based on risk level).

EVITA [30], HEAVENS [31] and SPMT [32] are automotive-specific frameworks that focus on risk assessment and analysis. These three frameworks provide methods for evaluating security threats by considering factors about the attack or threat, such as the required expertise of the attacker, necessary tools, and window of opportunity, and the impact of the threat, e. g., financial, safety and privacy impact. The outcome of such analyses is a classification of the identified threats showing their severity and likelihood. Yet, the specific aspect they present differs to some degree: EVITA and SPMT describe the risk of an attack using risk levels whereas HEAVENS results in security levels describing the required level of protection for each threat/asset combination. A detailed review of a variety of proposed TARA methods in regards to the applicability in the automotive context can be found in Macher et al. [33].

Figure 4 illustrates the steps when analysing a system and performing a TARA which results in a classification in form of levels. The item referred to in this figure can be a group of ECUs or a single ECU responsible to provide a certain function in the vehicle, e. g., braking and cruise control. A formal definition of item can be found in ISO 26262, that is *"[a] system or combination of systems [...] that implements a function or part of a function at vehicle level"* [34, p.16]. However, some threat analysis techniques (e.g., [30, 31]) identify assets, i. e., "anything [with] value to the organisation" [35], which are subject to security threats.

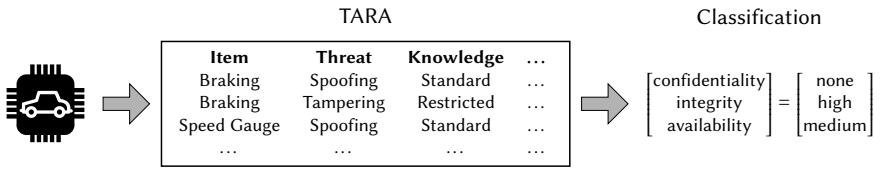


Figure 4: Steps when analysing a system and performing a TARA resulting in a classification.

Classification of security demands. There exist several different cyber security-related frameworks for the automotive domain [30–32, 36] and standards for other domains [37, 38] which classify the demands differently, i. e., the number of levels they suggest varies. The proposed levels range from four levels (*none, low, medium, high*) to up to eight levels depending on the framework in use. It should be also noted that a classification in levels can be used in a different context as well, for example, the use of levels in Common Criteria [38] and TAL [36]. Common Criteria's Evaluation Assurance Levels (EALs) show the requirements needed to be fulfilled when certifying the security of an IT system or product.

In addition to the varying number of risk and security levels in proposed frameworks, e. g., EVITA [30] (automotive) and IEC 62443 [37] (industrial communication networks), these two frameworks use a vector consisting of four respectively seven elements describing the demands for specific risks or security properties. The lack of consensus shown by the differences in the way how security frameworks and standards classify cyber security demands or the implied risk demonstrates that security is more complex and may need to be adapted for each domain. Safety standards from the automotive [34], avionics [39] and railway [40] domains on the other hand agree on having five levels (including *none/QM*).

The alignment to the functional safety standard ISO 26262 when structuring security demands is of particular interest in the automotive domain as it is widely accepted and implemented in the organisations. Functional safety is defined as the *"absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems"* [34, p.14]. This, however, is significantly different to cyber security which deals with intelligent individuals or organisations who can create complicated series of events that would otherwise never occur at random with the aim to steal private information, harm drivers and passengers, or disrupt our society.

Deriving security requirements. After assessing and rating the threats to the automotive system, high-level security requirements or directives need to be defined for each item respectively asset, aiming at eliminating or reducing the likelihood of the identified threats to happen.

The recommended method for deriving cyber security requirements based on the outcome of a TARA varies in each methodology. EVITA [30] and SPMT [32] both suggest ranking the threats according to their associated risk level and to perform an attack tree analysis where also the number of appearances of the sub-goals is considered. Attack trees [41] have the *ultimate goal* of the attacker in the root node and *sub-goals*, which are the necessary steps to achieve this goal, as leaf nodes. As

for many analyses, it should be noted that the outcome of the attack tree analysis also strongly depends on the level of detail with which the analysis is conducted. HEAVENS [31] follows a different approach which is similar to ISO 26262 which starts by defining *high-level security requirements* for each threat/asset pair and further defines as part of the development phase *technical security requirements* followed by concrete *hardware* and *software security requirements*. Clearly, these methods require much expertise and discussions between the practitioners on how each risk respectively threat is dealt with.

The vector representation describing the demands for certain risks or security attributes some frameworks [30, 32] and standards [37, 42] propose, aims at further assisting practitioners in choosing appropriate countermeasures for threats. NIST SP 800-53B [43] and IEC 62443 [37] go a step further and map security controls and mechanisms to security demands/levels. NIST FIPS PUB 199 [42] proposed already in 2004 to organise security demands in security categories, a vector consisting of three elements, i. e., the security objectives confidentiality, integrity and availability, each describing the impact (*none, low, moderate, high*). NIST also developed SP 800-53 [44] which identifies security and privacy controls for information systems and organisations and categorises them in 20 families, for example, access control, and awareness and training. In SP 800-53B [43] NIST additionally defines four baselines, namely the privacy baseline and 3 security control baselines (*low, moderate, high*), which assign security controls to these baselines. IEC 62443 [37], a network and system security standard for industrial communication networks, also describes a vector comprising of seven foundational requirements, and defines system requirements for each foundational requirement by differentiating between five security levels including *none*. This specific mapping to security requirements supports system architects and designers already at an early stage, i. e., directly after defining the security demands such as through TARA, about what requirements need to be implemented to reduce the likelihood of certain threats.

The concept of bringing such a mapping of security controls to automotive systems was also investigated by the *Connected Vehicle Pilot Deployment Program Phase 1* [45] project which used the CIA categories and levels from NIST FIPS 199. However, they do not provide a fixed mapping to the security controls defined in NIST SP 800-53, partly due to the fact that many NIST SP 800-53 controls are on an organisation-level.

Given the limitations and weaknesses of previous methods, we propose a structure to describe the security demands for automotive systems. This classification can be used to present the results of a TARA method such as HEAVENS (Chapter A). Moreover, we continue with this classification and identify suitable security requirements and mechanisms based on the proposed structure in order to provide practitioners with a selection of required mechanisms (Chapter B).

2.2 Cyber Security and Resilience Techniques

Security is often referred to as focusing on the three fundamental attributes, i. e., confidentiality, integrity and availability, whereas dependability is *"the ability to avoid service failures that are more frequent and more severe than acceptable"* [46, p. 13]. Therefore, dependability includes next to safety also other attributes, i. e., availability, reliability, integrity and maintainability [46]. Laprie [47] further elaborates on the need in ubiquitous systems to maintain dependability in spite of continuous change, which leads to the definition of resilience, namely *"the persistence of dependability when facing changes."* [47, p. 1]. From a security perspective, cyber resilience is often referred to as the ability to withstand attacks, more precisely in the CyReV project [27] resilience in the automotive context is defined as the *"property of a system with the ability to maintain its intended operation in a dependable and secure way, possibly with degraded functionality, in the presence of faults and attacks"*.

The relationship between resilience, security, dependability and safety is illustrated in Figure 5. The authors in [48] further differentiate between *resilience* and *scalable resilience* respectively *long-term dependability and security* to highlight that changes, i. e., technological, functional and environmental, occur over time and thus require the system to be capable to evolve. This emphasis on long-term needs is also of significant importance in the automotive domain, where vehicles are expected to be operated for several years or even decades. Due to the relation between safety, security and resilience, we refer to security when referring to techniques that directly support one of the security attributes and we refer to cyber resilience for techniques that may support both dependability and security.

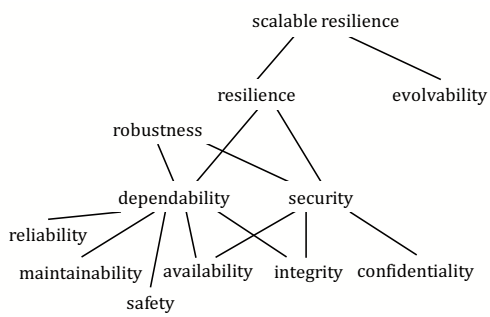


Figure 5: Relation between security, dependability and resilience attributes (adapted from [48])

Safety and security interplay. The alignment of security and safety techniques is not only crucial considering the importance of safety in the automotive domain, it is also important due to the interplay between safety and security techniques. Evaluating this interplay is not specifically addressed in this thesis, but the proposed methodology is aligned with ISO 26262 in order to ease such assessments during the concept and development phases. Some techniques, for instance, may complement each other, e. g., message counters to detect message loss (*safety*) and message

counters to detect replay attacks (*security*). Yet, safety and security mechanisms may also counteract one another if designed in isolation. For example, hardware redundancy as a safety feature may be exploited by an attacker as injecting malicious messages may result in both of the redundant systems believing that the other one is active. In respect to the interplay of safety and security mechanisms, a review of common security mechanisms and their implications to safety is provided in [49]. The authors review a selection of mechanisms and evaluate whether the identified mechanisms have a negative impact on dependability. Out of the 17 mechanisms 3 were identified to have a negative impact, i. e., access control, authentication control and encryption.

Categorisation. Efforts to identify and categorise security techniques have been made in various domains, most notably for IT organisations and systems (e. g., [44]). The common goal of these efforts is to support and guide system architects and designers in choosing appropriate techniques for the task at hand. A structured way to present security and resilience mechanisms and techniques is especially important after defining necessary security (and safety) requirements as a result of a TARA when generic requirements do not sufficiently cover the demands.

Techniques can be categorised in many ways. These categories are often based on the threat or attack they protect against or the security attribute they enforce. Choosing such a structure is natural when TARA methods already assess the security demands for specific threat categories such as Microsoft STRIDE [50], which details six threat types, i. e., spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege. When following more flexible methods such as attack tree analysis it may, however, be beneficial to provide a more general view on the techniques suitable for the automotive domain.

Frameworks categorising resilience techniques in particular often identify three or four strategies, yet emphasise different aspects. For example, NIST SP 800-160v2 [51] proposes a vocabulary for strategies (*anticipate, withstand, recover* and *adapt*) and further describes objectives and techniques to achieve organisation-wide cyber resilience. In comparison, Hukerikar and Engelmann [52] include three behavioural strategy patterns, i. e., *fault treatment, recovery* and *compensation*, following a categorisation based on fault, error, failure whereas Ratasich et al. [48] split resilience techniques into *detection, recovery or mitigation* and *long-term dependability and security* thus following a categorisation based on the attack progress. Clearly, the latter two approaches focus on supporting practitioners with a structure for selecting specific techniques on a system level whereas NIST SP 800-160v2 concentrates on a broader context, i. e., organization-wide cyber resilience.

With focus on the automotive domain, we identify and categorise security mechanisms based on an analysis of existing security standards and guidelines (Chapter B); and propose a taxonomy for resilience techniques as a result of a literature study (Chapter C). Ultimately, we analyse published attacks on automotive systems and propose a framework for designing secure and resilient systems (Chapter D).

2.3 In-vehicle Communication Security

Protecting the communication of the in-vehicle network from spoofing and tampering attacks requires message authentication. Due to the computational and bandwidth limitations of the in-vehicle network, many solutions for CAN message authentication have been proposed in literature [53–60]. The focus on CAN message authentication techniques is due to the limitations and therefore challenges in CAN, but also due to the fact that CAN is still used in some parts of the in-vehicle network as it is used to connect legacy systems and it is cheaper compared to Ethernet. This section gives details about the CAN bus and provides a brief introduction to message authentication and freshness.

Controller Area Network. CAN was initially developed by Robert Bosch GmbH in 1983 and later standardised in ISO 11898 [7]. The classical CAN bus allows bit rates up to 1 Mbit/s and a payload of 8 Bytes, however, adaptations exist that lift the bandwidth limitation by allowing higher bit rates and larger payload sizes, e.g., CAN FD. CAN was not developed with security in mind, CAN itself lacks basic security concepts, such as confidentiality, integrity, availability, authenticity and non-repudiation: *Confidentiality* cannot be ensured as CAN is a network bus where all messages are being broadcast and thus received by all entities connected to the bus. Confidentiality in CAN can be only achieved by message encryption on higher layers. *Integrity* is only provided by a checksum, which is not sufficient for protecting against attacks. *Availability* cannot be ensured as connected ECUs cannot be blocked from transmitting erroneous or high-priority messages. *Authenticity* is not provided by CAN since the senders of messages are unknown and able to send messages with any message identifier. *Non-repudiation* measures are also not included as they are typically combined with message authentication techniques [8].

A CAN base frame includes a 19 bit header which contains a start-of-frame bit, an 11 bit identifier, 3 control bits and 4 bits to describe the data length of the payload. The payload consists of up to 8 bytes and is followed by a 15 bit CRC, a delimiter, an acknowledgement bit and its delimiter and ends with 7 bits end-of-frame (EOF). The expected data in the payload, e. g., current speed and brake states, is solely described by the identifier field as the header does not contain any additional information about the sender of the frame.

Message Authentication Codes. Taking the small payload size of CAN frames into account it is evident that Message Authentication Codes (MACs) to ensure the authenticity of messages need to be truncated. Choosing the length of the truncated MAC depends on the required assurance against guessing attacks. The probability of a successful guessing attack is $2^{len_{MAC}}$ per verification attempt assuming the attacker randomly chooses a MAC. NIST SP 800-38B [61] describes CMAC, a MAC algorithm that is based on a symmetric key block cipher. NIST SP 800-38B discusses the selection of the MAC length in Appendix A.2 and emphasises to take the tradeoff between more robust and thus longer MACs and the performance and storage impact into account. In general, it can be noted that the necessary length greatly depends on the verification process, e. g., how often the verification process is allowed to fail due to a MAC mismatch and the timeframe in which the message is accepted. For this reason, it may be feasible to reduce the length of the MAC below 64 bit in cases

when the verification process is monitored by an Intrusion Detection System (IDS) and the validity of a message is short, e. g., only for a few seconds, as more recent messages are received.

It also needs to be decided whether to send the truncated MAC either (i) together with the data in one CAN frame or (ii) send a second CAN frame containing only the MAC. The former has the disadvantage that legacy ECUs which do not verify the authenticity of the frame need to be updated as the payload of a specific identifier, or signal, has changed. The latter approach considers this case and ensures that legacy ECUs do not need to be modified, yet a second identifier for each of the defined frames is needed to link the received MAC to its corresponding data frame. From a realtime-perspective, the delay caused by the need to wait for the second CAN frame carrying the MAC needs to be considered as well.

Freshness. Providing freshness for authenticated messages is important in order to be able to detect and prevent replay attacks. Freshness can be implemented in three different ways; by including (i) a timestamp, (ii) a counter, or (iii) a random nonce in each message when generating the MAC. Due to the bandwidth limitations of CAN, it is not feasible to include the entire counter or nonce with each message. Using a timestamp is in most cases not feasible as there is no synchronised global time due to bandwidth and ECU resource limitations. Hence, it is necessary to transmit only a truncated counter value and provide other means for synchronisation of the counter or nonce in cases an ECU gets out of sync. Most proposed message authentication solutions [53, 54, 56–59, 62] for the CAN bus focus on aspects such as key distribution, backwards compatibility, hardware and bandwidth limitations. They do not directly address the challenge of synchronising freshness counters.

Existing work on providing freshness to authenticated messages either propose to periodically synchronise the counter [55, 60] (e. g., every 50 ms [55]), follow a hardware-based approach [63] and thus requiring modifications of the CAN transceiver, or misuse header fields to accommodate a counter [64].

With these limitations in mind, we analyse an existing freshness solution for authenticated messages (i. e., AUTOSAR SecOC Profile 3) and propose an extension to overcome these identified limitations (Chapter E).

2.4 Vehicle Trust in VANETs

VANETs consist of vehicles directly communicating with each other (V2V) and to infrastructure (V2I) like Roadside Units (RSUs). This type of short-range communication paves the way for new use cases that aim at increasing traffic safety and efficiency [65] for two main reasons, (i) vehicles are able to receive information about objects and warnings which they cannot measure or detect with their own sensors (detection of *out-of-sight* objects); and (ii) interact with each other in a cooperative way to increase traffic efficiency, such as platooning and cooperative intersections. This section introduces the two technologies that can be used for V2X communication and gives a concise overview of trust and reputation models for cooperative vehicles as well as intrusion detection and the challenges still remaining.

V2X technologies. There are two distinct technologies for enabling V2X (V2V and V2I) communication, namely Cooperative-ITS (C-ITS) and Cellular-V2X (C-V2X). The former is based on IEEE 802.11-2016 OCB mode, previously known as IEEE 802.11p, and specifies the access layer for an ad hoc network (VANET) operated in the 5 GHz frequency band in ETSI ITS-G5 [66]. The upper layers are defined in ETSI ITS specifications in Europe and IEEE 1609 in the US. C-V2X, which is specified by the 3rd Generation Partnership Project (3GPP), on the other hand, extends cellular technologies, i. e., LTE and 5G, to accommodate V2X communication. 3GPP defines the "traditional" cellular network (Uu) and short-range communication (PC5) allowing direct communication between vehicles and RSUs. The security aspects of these specifications are reviewed in detail by Yoshizawa and Preneel [67]. In general, it can be noted that both technologies, C-ITS and C-V2X, rely on the upper layers to implement security mechanisms to protect the short distance communication.

ETSI TS 102 940 [68] specifies the security architecture for ITS and links to the relevant specifications that describe authentication and confidentiality requirements, and how certificates should be managed and verified in detail. ETSI ITS TR 102 893 [22] recommends complementing cryptographic measures with additional plausibility checks to verify that the received information is valid. Such plausibility checks have been proposed in literature in various ways, often referred to as *trust* or *reputation models*. Such models can be categorised into three groups, namely entity-oriented trust, data-orient trust and combined trust models [21]. *Entity-oriented* trust concentrates on modelling the trust in the surrounding vehicles. *Data-oriented* trust focuses on expressing the trust in certain information, e. g., a slippery road ahead warning message, received from several vehicles. *Combined* trust uses a trust model for evaluating the trust in peer vehicles by considering also its peer's trust evaluation [21].

Vehicle trust. Entity-oriented trust in the context of VANETs can be measured by participating vehicles through (i) validation of information with their own sensors or based on a model, and (ii) evaluating the compliance to interaction protocols and behaviour when performing cooperative actions. The validation of information in regards to correctness and accuracy is important for the *own* vehicle, further named *ego* vehicle, especially during cooperative scenarios like platooning where the vehicles need to heavily rely on the reported data. Vehicles can validate this information either with their own sensors, e. g., camera, radar and lidar, if the vehicle in question is captured by these sensors or they can rely on a non-linear filter applied only on the reported information, such as Kalman filters [69] and particle filters [70], to identify and approximate temporary inaccuracies of the reported information. The evaluation of the behaviour during cooperative scenarios is also vital for automated vehicles when making decisions about to what extent they cooperate with a certain vehicle. Such evaluations can mainly be performed by observing the cooperation of the vehicle in question with other vehicles and by the ego vehicle after cooperating with it. Furthermore, the results of such trust assessments can be either binary (*trust* or *distrust*) or more detailed using a trust rating, index or score, e. g., a numerical value within range $[0, 1]$.

Challenges in establishing trust. The challenges in evaluating the trust of vehicles are for the most part due to the nature of VANETs. Communication in VANETs is

typically temporary and can last from a few seconds, when a faster vehicle passes the ego vehicle, to hours, when platooning. Therefore, it is important that entity-oriented trust can be established quickly after encountering a vehicle. Moreover, a vehicle has limited resources to compute and store the results of these trust evaluations for every vehicle it encounters. For this reason, some solutions also propose ways to share trust evaluations with other vehicles. Examples of such techniques range from exchanging trust scores in the VANET [71, 72], using the infrastructure respectively the cloud to request and aggregate trust opinions [73, 74], to using blockchain to maintain a ledger of reported trust scores [75]. Privacy concerns and the use of pseudonym certificates which are regularly changed to maintain the privacy of a driver/vehicle are also challenging for deploying a trust model. For building trust or request historical trust scores using infrastructure, it is needed to identify the vehicles and therefore solutions involving certificate authorities who issue the pseudonym certificates to the vehicles would be needed to receive the correct trust scores. In addition to the aforementioned challenges, Hussain et al. [20] also identify the need for incentives for participating in such trust evaluations as well as auditing them.

Trust models. Modelling trust in surrounding vehicles is important for increasing resilience to inaccuracies in reported information and faults caused by other vehicles, as automated vehicles can include this information in their decision-making. Trust evaluations and their resulting trust scores can be used to identify malicious vehicles that drop or manipulate messages they forward or generate, and vehicles that behave differently in certain driving situations, e. g., when the road is slippery, or during cooperative driving scenarios. Many proposed models [71, 73, 76, 77] focus on identifying malicious behaviour by evaluating the validity of event messages and/or network behaviour such as message drop rate and packet forward delay. Interested readers may refer to Hussain et al. [20] for a detailed review of existing trust management schemes. Other solutions modelling the vehicle dynamics or validating sensor data [70, 78, 79], e. g., by using sensor fusion, concentrate on vehicle driving behaviour and correctness of the provided sensor information. The information gained from such solutions is also important from a cyber security and resilience stand since maliciously behaving vehicles and vehicles experiencing software or hardware faults may also show an altered, not expected behaviour.

Intrusion detection. The aim of proposed solutions investigating anomalies and intrusions caused by vehicles in VANETs varies depending on the focus of the research. Trust-centred solutions concentrate on increasing the awareness of automated and autonomous vehicles in order to increase resilience towards attacks and faults. Security-focused research concentrating on IDSs on the other hand deals mainly with detecting attacks and consequently raising an alert. IDSs for VANETs, however, focus, similar to trust-based solutions, on network-related characteristics and ways to distribute this knowledge. Sharma and Kaul [80] provide a survey on IDSs in VANETs and also conclude that many IDSs consider only one or two different attacks and that there is a need for a more generalised IDS which is able to detect a variety of attacks.

In summary, combining the different approaches that evaluate trust in individual vehicles can potentially enable new ways to detect anomalies caused by attackers or

random software or hardware faults based on (i) the participation in the VANET as a routing node (e. g., dropping or forwarding manipulated packets); (ii) the correctness of information about events (e. g., manipulated or wrong warning messages); (iii) the accuracy of reported sensor data (e. g., speed, acceleration, geographic location, and heading); and (iv) the behaviour during cooperative scenarios (e. g., platooning).

We propose a trust model which, in contrast to previous work, combines the evaluation of the behaviour of surrounding vehicles and their cooperation with other vehicles (Chapters F and G). Furthermore, we suggest a framework for identifying anomalies using such a trust evaluation and combine it with an analysis of cloud data in order to detect such anomalies in a vehicle fleet (Chapter H).

3 Thesis Objectives

This thesis investigates how to design secure automotive systems, i. e., by studying security and resilience techniques, securing the in-vehicle communication, and by assessing the information received in the VANET and the behaviour of other vehicles. By not only focusing on security but also including resilience, we take the capabilities and needs of future vehicles, which drive autonomously, into account. This thesis particularly concentrates on the following research questions:

RQ1 How to express security demands and requirements when designing and developing automotive systems? Can specific security requirements be generalised to cover basic security needs?

RQ2 What are suitable security and resilience techniques for automotive systems? How can these techniques be organised to aid practitioners in selecting them?

RQ3 How to protect communication in the in-vehicle network? What are the limitations and design considerations when introducing freshness in authenticated messages and how can we overcome them?

RQ4 How can deviations from the intended functionality of a vehicle be detected even when it is compromised and thus unreliable when assessing its own state?

RQ1. Describing the demands for security in form of levels or classes is important for communicating the overall security needs of an item, i. e., a function, device or a group of functions, with all parties involved in the design and continued maintenance of a vehicle. *RQ1* also emphasises the need for investigating what this categorisation should look like and whether specific security requirements can be solely derived based on this classification. This thesis investigates a structure for representing security demands in the automotive domain based on a study on existing standards and frameworks also from other domains. In addition, generic security requirements and mechanisms have been identified and mapped to the proposed classification.

RQ2. During the design and development of automotive systems it is of utmost importance that practitioners are able to choose appropriate techniques and mechanisms to build systems that withstand and react to attacks accordingly. This thesis addresses *RQ2* from three different angles, by focusing first only on security, then on resilience techniques and ultimately on combining these techniques in one framework.

RQ3. The freshness of messages is particularly important when protecting against replay attacks. Identifying that the received message is *old* and thus not valid anymore is vital when safety-related signals are exchanged on the internal network, e.g., sending a previously recorded brake signal could cause serious harm. Hence, authenticated messages need to include a counter, nonce, or timestamp. Since ECUs likely do not have a synchronised time, it is necessary to use counter-based solutions. These solutions, however, have the disadvantage that either the entire counter value or nonce has to be transmitted with each message or a synchronisation of the message counter needs to be in place. The work presented in this thesis focuses on the use of counter synchronisation which works with networks that have already high load and low bandwidth, such as CAN.

RQ4. Information received via V2V communication is cryptographically secured by means of message authentication. However, vehicles may provide inaccurate or faulty information due to internal randomly occurring faults which the vehicle may not be aware of. Furthermore, the vehicle and its firmware may have been subject to manipulation by an unauthorised entity, such as the owner, driver, or an attacker, which causes malicious, unintended or unwanted behaviour. Identifying such anomalies and attacks is important for resilient vehicles as it allows them to include this knowledge in their decision-making process and to mitigate and recover from ongoing attacks.

Based on these research questions, the remaining chapters in this thesis are split into Part II *Generic Security Requirements and Identification of Suitable Techniques* dealing with the identification of security and resilience techniques; and Part III *Design and Evaluation of Security and Resilience Techniques* presenting specific security and resilience methods. The mapping of the research questions to the respective chapters in this thesis is shown in Table 1.

Table 1: Research questions addressed in each chapter.

	II. Generic security requirements and identification of suitable techniques				III. Design and evaluation of security and resilience techniques		
	Cyber security requirements		Cyber security and resilience techniques		In-vehicle security	Vehicle trust in VANETs	
	Chapter A	Chapter B	Chapter C	Chapter D	Chapter E	Chapter F & G	Chapter H
RQ1	●	●	○	○	○	○	○
RQ2	○	●	●	●	○	○	○
RQ3	○	○	○	○	●	○	○
RQ4	○	○	○	○	○	●	●

In brief, **Chapter A** presents the challenges in the automotive domain and investigates how security demands and security levels should be structured to allow a better understanding between different entities within and outside the organisation. **Chapters B, C and D** each identify suitable mechanisms and techniques to provide cyber security and/or resilience for vehicles using different approaches, which are (i) an analysis of standards and guidelines, (ii) a literature study and (iii) techniques based on the analysis of disclosed automotive attacks. **Chapter E** analyses an existing solution that provides freshness to authenticated messages to detect replay attacks and further proposes an extension to overcome the identified limitations when synchronising the freshness value. **Chapter F** presents the system architecture of a cooperative, V2X-enabled vehicle and an overview of the research performed with it. This architecture is also used in **Chapter G** to develop a trust model to enable the automated vehicle to make decisions by including the evaluation of V2V communication. **Chapter H** defines a framework for anomaly detection in a fleet of vehicles by combining the peer evaluation using vehicle trust with the subsequent analysis in the cloud.

4 Thesis Contributions

This thesis contributes to the research questions raised in Section 3 addressing the needs for security and resilience in automotive systems involving functions using the in-vehicle network, VANET and the cloud. The structure of the following chapters of this thesis and the research questions they address are described in Table 1.

4.1 Generic Security Requirements and Identification of Suitable Techniques

To identify and subsequently categorise security and resilience techniques, it is essential to understand the specific challenges in automotive systems and how security demands in form of security levels can be structured. The identification and categorisation of security and resilience techniques is needed to allow designers to choose and make an informed decision from a set of mechanisms applicable to automotive systems. **Chapter A** presents a general overview of the challenges in the automotive domain and how to express and represent security demands. **Chapter B** identifies generic security requirements and explores how these can be mapped to security levels resulting from a TARA. In **Chapter C**, we perform a literature study and propose a taxonomy for resilience mechanisms for automotive systems. **Chapter D** subsequently combines security and resilience mechanisms based on a detailed analysis of disclosed attacks.

Chapter A: Open Problems when Mapping Automotive Security Levels to System Requirements [81]

Communicating safety demands and requirements is well established in the automotive domain; mainly due to the broad acceptance of ISO 26262 [6]. Therefore, existing research has focused on ways how to extend or align these existing methods from the

safety domain with approaches from the security domain. Methods, e. g., [30, 31, 82], concentrate on security assessments that are aligned to the Hazard Analysis and Risk Assessment (HARA), a method to identify and categorise hazards.

This chapter provides an analysis of standards and frameworks for the automotive and other industrial domains concerning their methodology on how security respectively safety is classified. Describing the security demands in form of classes or levels is necessary to place an emphasis on the need for security measures for a certain function or ECU. We begin by highlighting the challenges that the automotive domain faces when designing and developing secure systems. Second, we study how selected standards and frameworks structure the chosen classifier for their corresponding domain. Third, we propose a structure for automotive security levels that considers the identified challenges which are specific to the automotive domain. The structure is a vector consisting of six elements where each element describes the demands for a specific security attribute, i. e., authenticity, integrity, non-repudiation, confidentiality, availability and authorisation, using 5 levels.

Statement of contributions. This is a joint work with my supervisor Tomas Olovsson. I contributed as the lead for the idea, performing the analysis and writing of the manuscript.

Chapter B: Towards a Standardized Mapping from Automotive Security Levels to Security Mechanisms [83]

Proposed techniques for threat assessment, specifically TARA, focus on ways to evaluate the risk or need for protection and therefore require to manually derive security requirements for each item that is being assessed (e. g., [30–32, 82]). Standards and frameworks from other domains have also looked into identifying suitable security mechanisms and directly linking them to the security demands in order to provide designers with the minimal requirements for the item under assessment. Examples of such mappings are NIST SP 800-53 [44], with focus on information systems, and IEC 62443 [37] which concentrates on industrial communication networks.

In this work, we continue at the point of having performed a TARA resulting in a set of Security Levels (SLs) for each identified asset, which can be a function, a vehicle ECU, or even a network segment. The proposed framework guides designers and engineers in identifying necessary security mechanisms to be implemented based on a mapping from SLs to a list of mandatory security mechanisms. We have identified appropriate security mechanisms suitable for automotive systems and assigned them to particular SLs to provide a strict rule-set that is required to fulfil basic security demands. This approach allows designers to focus on application-specific requirements, supports them to identify conflicts and dependencies between safety and security requirements in an early stage of development and provides a common ground when communicating security requirements between different actors involved in the development. Furthermore, the framework has been verified together with a vehicle manufacturer based on a use case showing how this framework can be applied.

Statement of contributions. This is a joint work with my supervisor Tomas Olovsson. I contributed as the lead for the idea, performing the analysis and writing of the manuscript.

Chapter C: REMIND: A Framework for the Resilient Design of Automotive Systems [84]

Resilience of automotive systems is required in order to cope with diverse and newly emerging attacks that make use of the advances in communication and functionality. Automotive systems need to maintain the intended functionality, even if degraded, to ensure the safety of the passengers and the surrounding environment. Research in identifying and categorising resilience techniques has been performed in areas such as cloud computing [52], fog computing [85] and cyber-physical systems [48].

In this work, we review and analyse scientific literature on resilience techniques, fault tolerance, and dependability. As a result, we present the REMIND resilience framework supporting the design of resilient vehicles by (i) identifying techniques for attack detection, mitigation, recovery and resilience endurance; (ii) organising these techniques into a taxonomy to guide designers in choosing the needed technique for the task at hand; (iii) providing guidelines describing how the proposed taxonomy can be applied against common security threats; and (iv) discussing the trade-offs when implementing techniques identified in REMIND.

Statement of contributions. This is a joint work with the co-authors. I contributed as the lead for the idea, performed the literature review, was the lead for designing the taxonomy for resilient techniques (REMIND) and collaborated with all other authors. Rodi Jolak was primarily responsible for the guidelines for using REMIND, which are presented in Appendix C.A. Kim Strandberg contributed particularly to the attack model and asset identification.

Chapter D: Resilient Shield: Reinforcing the Resilience of Vehicles Against Security Threats [86]

Existing work on securing vehicles focuses on providing frameworks that help designers and developers in identifying the necessary mechanisms to mitigate various attack scenarios. Microsoft STRIDE [50], for instance, provides a tool for threat modelling, HEAVENS [31] supports developers in defining security objectives based on their proposed TARA and other work, such as Sommer et al. [87] focus on a taxonomy for attacks against automotive systems.

This chapter combines security and resilience techniques needed in automotive systems in one framework. For the proposed framework, we apply the SPMT methodology on systematically identified attacks in order to derive security guidelines as well as detailed directives focusing on security and resilience. We further map the potential threat actors to the assets exposed by each attack and show which security and resilience techniques can be deployed to mitigate them. The resulting framework, named Resilient Shield, builds the base for designing secure and resilient systems, yet allows to be easily extended in the presence of novel attacks.

Statement of contributions. This is a joint work with the main and co-authors. Kim Strandberg contributed as lead with the idea. My contributions are the shared effort with the main author to find and review attacks. I was the lead for the initial attack assessment, assignment of mitigation techniques and automotive assets. We contributed equally to identifying the detailed directives of which Resilient Shield is composed.

4.2 Design and Evaluation of Security and Resilience Techniques

This thesis includes three mechanisms to provide security and resilience for automotive systems. **Chapter E** investigates the shortcomings of a proposed solution for counter-based message authentication in the in-vehicle communication and shows how to overcome them. **Chapter F** describes the vehicle architecture of a cooperative, V2X-enabled vehicle and presents the evaluation results when tested in the field. Based on this architecture, **Chapter G** details a trust model which allows automated vehicles to include the reliability of the information received via V2V communication in their decision-making. **Chapter H** takes trust evaluations one step further and investigates how they can be used in combination with data in the cloud to detect and identify anomalies in vehicles on a fleet level.

Chapter E: Extending AUTOSAR's Counter-based Solution for Freshness of Authenticated Messages in Vehicles [88]

To protect against replay attacks, it is needed to ensure that the messages sent on the network are newly generated. This security property called *freshness* can be achieved by adding a timestamp, nonce, or counter when authenticating messages. AUTOSAR, an open system platform for vehicles, includes two so-called SecOC Profiles that provide freshness; one uses a single counter and the other one, SecOC Profile 3, describes the structure for a counter and means to synchronise it between ECUs. The synchronisation of the counter or Freshness Value (FV) is necessary in order to avoid sending the complete FV with each message as the CAN bus is already highly utilised. Proposed solutions for counter-based message authentication solutions designed for the CAN bus either require specific hardware [63], propose periodic synchronisations [55, 60], misuse other fields in the CAN frame [64], lack details on how a synchronisation can be achieved [89], or leave the resynchronisation to the underlying protocol [90].

In this work, we focus on SecOC Profile 3 and analyse its usability and ability to synchronise the FV. We discuss possible designs and limitations for such a mechanism deployed in vehicles and consequently propose an extension that mitigates the identified shortcomings. Our novel approach allows faster resynchronisation and requires fewer synchronisation messages to be transmitted. Furthermore, the proposed extension has been implemented on a testbed and evaluated in terms of flexibility, time for resynchronisation and additional bus load with favourable results.

Statement of contributions. This is a joint work with Christian Sandberg and my supervisor Tomas Olovsson. I contributed as the lead for the idea, performing the analysis and writing of the manuscript.

Chapter F: Team Halmstad Approach to Cooperative Driving in the Grand Cooperative Driving Challenge 2016 [91]

Cooperative driving enables the sensing of out-of-sight objects, e. g., receive road warnings, and allows vehicles to solve tasks more efficiently by interacting with each other. This work provides an overview of the control and communication system developed in the course of the Grand Cooperative Driving Challenge 2016

with the focus on design choices made to fulfil the competition's requirements, and post-competition evaluations. The competition particularly focused on evaluating two scenarios, i. e., a cooperative lane merge due to road work on the left lane, and a cooperative intersection.

In this chapter, we present our implementation of a Cooperative Adaptive Cruise Control (CACC), our solution for communication and logging, and our approach for high-level decision making.

Statement of contributions. This is a joint work with all authors. I contributed with the design and implementation of the perception and sensor fusion module and the trust system and equally contributed with two other authors in the design and implementation of the high-level system control. In addition, I contributed to the design of the vehicle architecture.

Chapter G: Modelling the Level of Trust in a Cooperative Automated Vehicle Control System [92]

Cooperative scenarios can increase safety and can lead to an increased efficiency in power or fuel consumption, e. g., when platooning [3]. Relying completely on cryptographic solutions that provide authenticity, integrity and, when needed, confidentiality, however, is not sufficient when processing information received from other vehicles. It is required to ensure that the received information is reliable, as the safety of the passengers is at risk. The need for a trust-based evaluation for decision-making has been also identified in literature [20, 21].

In this chapter, we explore and propose how to include trust or the reputation of other vehicles in the decision-making process of an automated vehicle when performing cooperative actions. We present a trust model which includes four trust scores or indices in its decision-making, namely the trust (i) in the ego vehicle, (ii) in the vehicle in front, (iii) in surrounding vehicles and (iv) in the environment. Furthermore, we demonstrate how and why the trust indices change using data of real V2V interactions from the Grand Cooperative Driving Challenge (GCDC) 2016.

Statement of contributions. This is a joint work with my Master thesis supervisor Cristofer Englund. We contributed equally to the idea and I was lead for the design and implementation of the trust system and writing of the manuscript.

Chapter H: V2C: A Trust-Based Vehicle to Cloud Anomaly Detection Framework for Automotive Systems [93]

Vehicle trust is primarily proposed in literature to overcome the difficulty of relying on information received via V2V communication when interacting with other automated vehicles. Related work also identified the potential of vehicle trust in combination with intrusion detection, however, these solutions are either focusing on collaborating with trusted vehicles and including the results in the own vehicle's decision-making [94] or only consider packet header information in their intrusion detection system [95, 96].

This chapter proposes an anomaly detection framework that utilises the trust scores computed by individual vehicles based on V2V interactions by combining

them with a subsequent analysis in the cloud. The presented *V2C Anomaly Detection* framework consists of four modules dividing the tasks into (i) individual assessments of neighbouring vehicles resulting in a trust score; (ii) aggregation of these individual assessments to one trust score per vehicle; (iii) detection of anomalies or changes in the trust score over time; (iv) further analysis using data available in the cloud to also detect similar anomalies in a vehicle fleet. The advantage of using trust evaluations for detecting anomalies is twofold. First, these vehicle assessments are, unlike IDSs, performed by other vehicles and not by the system itself which is important because the vehicle itself may not be aware of the fault or the IDS may also be compromised. Second, this framework is scalable as the computational costs for observing the trust score for each vehicle and triggering a detailed analysis only when changes in the score are detected, requires less resources than triggering a comprehensive analysis with each newly uploaded event to the cloud. For each module comprising this framework we define their requirements, show how the identified threats can be detected, provide detailed discussions about suitable techniques and propose modifications if necessary. Furthermore, we identify attack scenarios which such a framework can detect and discuss its applicability in a detailed discussion based on a use case.

Statement of contributions. This is a joint work with my supervisors Tomas Olovsson and Magnus Almgren. I contributed as the lead for the idea, development of the framework and writing of the manuscript.

5 Conclusion

Cyber security and resilience have become a necessity for automated vehicles as their capabilities and connectivity increased tremendously in the past years making them attractive targets for attackers due to their influence on our society and the data they produce. This thesis concentrates on several aspects necessary to secure these vehicles, i. e., guide practitioners to derive requirements and choose appropriate techniques, and help with the design of specific techniques suited for vehicles. Furthermore, by expanding our research to include also cyber resilience adds the emphasis that systems need to be able to reconfigure and recover by themselves in the presence of faults and attacks. The following are the main contributions of this thesis with respect to the research questions defined in Section 3.

Chapters A, B, C and D focus on how to define generic requirements and the identification of suitable techniques and thereby contribute to *RQ1* and *RQ2*. In detail, **Chapter A** explores the unique challenges when securing vehicles and how security demands should be structured in order to provide practitioners from different organisations with a common language for describing security needs. **Chapter B** continues with using the proposed structure for classifying security demands in form of security levels, identifies generic security requirements and suggests a mapping based on these demands. **Chapter C** provides a taxonomy for resilience techniques and performs a literature study to identify techniques suitable for the automotive domain. **Chapter D** analyses disclosed attacks on vehicles to aid practitioners in selecting resilience and security techniques based on automotive assets.

Chapters E, F, G and H concentrate on the design and evaluation of specific security or resilience techniques, i. e., *RQ3* and *RQ4*. **Chapter E** concentrates on securing the in-vehicle communication by extending an existing freshness mechanism for authenticated messages to allow more flexibility and a faster resynchronisation when only truncated freshness values are exchanged. **Chapter F** describes the system architecture of an experimental cooperative vehicle which is further used in **Chapter G** to evaluate the proposed trust model which is applied to the vehicle itself and the vehicles it interacts with to improve resilience when driving. **Chapter H** presents a framework making use of vehicle trust, which is based on evaluating the communication and interaction with other vehicles, to detect anomalies and consequently trigger an analysis in the cloud using also the available cloud data in order to (i) find vehicles in the fleet with similar behaviour; and (ii) to identify the cause of the anomaly which can be a failure or an attack.

Interesting challenges and research questions as future work include further investigations on how to design and develop resilient vehicles. Cyber resilience in the automotive context is particularly interesting as it requires an even closer integration of security and safety as well as challenges involving the evolvability of a system and its self-awareness.

Bibliography

[1] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.

[2] K. Sjöberg, P. Andres, T. Buburuzan, and A. Brakemeier, "Cooperative intelligent transport systems in Europe: Current deployment status and outlook," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 89–97, 2017.

[3] A. Davila and M. Nombela, "Platooning - safe and eco-friendly mobility." *SAE Technical Paper 2012-01-0488*, 2012.

[4] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 447–462.

[5] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. USA: USENIX Association, 2011, p. 6.

[6] ISO, "ISO 26262:2011 Road vehicles – functional safety," International Organization for Standardization, Standard, 2011.

[7] ——, "ISO 11898-1:2015 Road vehicles – controller area network (CAN) – part 1: Data link layer and physical signalling," International Organization for Standardization (ISO), Standard, 2015.

[8] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks – practical examples and selected short-term countermeasures," in *Computer Safety, Reliability, and Security*, M. D. Harrison and M.-A. Sujan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 235–248.

[9] L. Yue, Z. Li, T. Yin, and C. Zhang, "CANCloak: Deceiving two ECUs with one frame," in *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, vol. 2021, 2021, p. 25.

[10] P. Carsten, T. R. Andel, M. Yampolskiy, and J. T. McDonald, "In-vehicle networks: Attacks, vulnerabilities, and proposed solutions," in *Proceedings of*

the 10th Annual Cyber and Information Security Research Conference, ser. CISR '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2746266.2746267

[11] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[12] P. Murvay and B. Groza, "DoS attacks on controller area networks by fault injections from the software layer," *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017.

[13] R. N. Charette, "How software is eating the car," June 2021, (Accessed: 2021-06-17). [Online]. Available: https://spectrum.ieee.org/cars-that-think/transportation/advanced-cars/software-eating-car

[14] SAE, "SAE J3061: SURFACE VEHICLE RECOMMENDED PRACTICE - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," SAE International, Recommended Practice, 2016.

[15] ISO/SAE, "ISO/SAE 21434 Road vehicles – cybersecurity engineering," International Organization for Standardization, Standard, 2020.

[16] R. G. Engoulou, M. Bellaïche, S. Pierre, and A. Quintero, "VANET security surveys," *Computer Communications*, vol. 44, pp. 1–13, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366414000863

[17] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "VANet security challenges and solutions: A survey," *Vehicular Communications*, vol. 7, pp. 7–20, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209616301231

[18] M. Botte, L. Pariota, L. D'Acierno, and G. N. Bifulco, "An overview of cooperative driving in the European Union: Policies and practices," *Electronics*, vol. 8, no. 6, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/6/616

[19] C. A. Kerrache, C. T. Calafate, J. Cano, N. Lagraa, and P. Manzoni, "Trust management for vehicular networks: An adversary-oriented overview," *IEEE Access*, vol. 4, pp. 9293–9307, 2016.

[20] R. Hussain, J. Lee, and S. Zeadally, "Trust in VANET: A survey of current solutions and future research opportunities," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–19, 2020.

[21] J. Zhang, "A survey on trust management for VANETs," *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pp. 105–112, 2011.

[22] ETSI, "ETSI ITS TR 102 893 v1.2.1 – threat, vulnerability and risk analysis (TVRA)," European Telecommunication Standards Institute, Technical Report, 2017.

[23] T. Fox-Brewster and Forbes, "BMW update kills bug in 2.2 million cars that left doors wide open to hackers," Feb 2015, (Accessed: 2021-09-11). [Online]. Available: http://www.forbes.com/sites/thomasbrewster/2015/02/02/bmw-door-hacking

[24] Vulnerability Lab, "BMW ConnectedDrive - (update) VIN session vulnerability," July 2016, (Accessed: 2021-09-11). [Online]. Available: https://www.vulnerability-lab.com/get_content.php?id=1736

[25] ——, "BMW - (token) client side cross site scripting vulnerability," July 2016, (Accessed: 2021-09-11). [Online]. Available: https://www.vulnerability-lab.com/get_content.php?id=1737

[26] T. Hunt, "Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs," https://www.troyhunt.com/controlling-vehicle-features-of-nissan/, 2016, (Accessed: 2020-09-11).

[27] Vinnova, "CyReV (phase1) – cyber resilience for vehicles - cybersecurity for automotive systems in a changing environment," https://www.vinnova.se/en/p/cyrev-phase1---cyber-resilience-for-vehicles---cybersecurity-for-automotive-systems-in-a-changing-environment, 2018, (Accessed: 2021-09-11).

[28] ——, "HoliSec: Holistic approach to improve data security," https://www.vinnova.se/en/p/holisec-holistic-approach-to-improve-data-security/, 2015, (Accessed: 2021-09-11).

[29] A. Yadav and C. Sandberg. (2018) Holisec reference architecture. (Accessed: 2018-04-10). [Online]. Available: http://autosec.se/wp-content/uploads/2018/04/HOLISEC_D4.1.3_v1.0.pdf

[30] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weylr, "Security requirements for automotive on-board networks," in *9th International Conference on Intelligent Transport Systems Telecommunications, (ITST)*. Institute of Electrical and Electronics Engineers (IEEE), 2009.

[31] M. M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson, "A risk assessment framework for automotive embedded systems," in *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security - CPSS 16*. Association for Computing Machinery (ACM), 2016.

[32] K. Strandberg, T. Olovsson, and E. Jonsson, "Securing the connected car: A security-enhancement methodology," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 56–65, 2018.

[33] G. Macher, E. Armengaud, E. Brenner, and C. Kreiner, "A review of threat analysis and risk assessment methods in the automotive context," in *Computer Safety, Reliability, and Security*, A. Skavhaug, J. Guiochet, and F. Bitsch, Eds. Cham: Springer International Publishing, 2016, pp. 130–141.

[34] ISO, "ISO 26262-1:2018 Road vehicles – functional safety – part 1: Vocabulary," International Organization for Standardization, Standard, 2011.

[35] ENISA, "Glossary – published under risk management," 2021, (Accessed: 2021-03-15). [Online]. Available: https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/glossary

[36] A. Kiening, D. Angermeier, H. Seudie, T. Stodart, and M. Wolf, "Trust assurance levels of cybercars in V2X communication," in *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles - CyCAR 13*. Association for Computing Machinery (ACM), 2013.

[37] ISA/IEC, "IEC 62443 – Industrial communication networks - network and system security," International Electrotechnical Comission, Standard, 2013.

[38] "CC v3.1. Common Criteria for information technology security evaluation," Common Criteria, Standard, 2017.

[39] "RTCA DO-178 – software considerations in airborne systems and equipment certification," RTCA and EUROCAE, Standard, 2011.

[40] European Standards, "EN 50129 – Railway applications - communication, signalling and processing systems - safety related electronic systems for signalling," EN, Standard, 2019.

[41] B. Schneier, "Modeling security threats," *Dr. Dobb's journal*, vol. 24, no. 12, 1999, (Accessed: 2021-03-16). [Online]. Available: https://www.schneier.com/academic/archives/1999/12/attack_trees.html

[42] "NIST FIPS PUB 199 – Standards for Security Categorization of Federal Information and Information Systems," National Institute of Standards and Technology, Standard, 2004. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.199

[43] "NIST SP 800-53B – Control baselines for information systems and organizations," National Institute of Standards and Technology, Special Publication, 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-53B

[44] "NIST SP 800-53r5 – Security and privacy controls for information systems and organizations," National Institute of Standards and Technology, Special Publication, 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-53r5

[45] S. Galgano, M. Talas, W. Whyte, J. Petit, D. Benevelli, R. Rausch, and S. Sim, "Connected vehicles pilot deployment phase 1 – Security management operating concept – New York City," U.S. Department of Transportation, Tech. Rep. FHWA-JPO-16-300, 2016.

[46] A. Avizienis, J. . Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[47] J.-C. Laprie, "From dependability to resilience," in *38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks*, 2008, pp. G8–G9.

[48] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci, "A roadmap toward the resilient internet of things for cyber-physical systems," *IEEE Access*, vol. 7, pp. 13 260–13 283, 2019.

[49] B. Sangchoolie, P. Folkesson, P. Kleberger, and J. Vinter, "Analysis of cyber-security mechanisms with respect to dependability and security attributes," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2020, pp. 94–101.

[50] Microsoft Corporation, "The STRIDE threat model," 2005, (Accessed: 2017-02-23). [Online]. Available: https://msdn.microsoft.com/en-us/library/ee823878.aspx

[51] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid, "Developing cyber resilient systems: A systems security engineering approach," National Institute of Standards and Technology, Gaithersburg, MD, Special Publication NIST SP 800-160v2, Nov. 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v2.pdf

[52] S. Hukerikar and C. Engelmann, "Resilience design patterns: A structured approach to resilience at extreme scale," *arXiv preprint arXiv:1708.07422*, 2017.

[53] B. Groza, S. Murvay, A. van Herrewege, and I. Verbauwhede, "LiBrA-CAN: A lightweight broadcast authentication protocol for controller area networks," in *Cryptology and Network Security*, J. Pieprzyk, A.-R. Sadeghi, and M. Manulis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 185–200.

[54] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "CaCAN-centralized authentication system in CAN (controller area network)," in *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*, 2014.

[55] S. Nürnberger and C. Rossow, "vatiCAN - Vetted, Authenticated CAN Bus," in *Cryptographic Hardware and Embedded Systems – CHES 2016*, B. Gierlichs and A. Y. Poschmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 106–124.

[56] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann, "Car2X communication: Securing the last meter - a cost-effective approach for ensuring trust in Car2X applications using in-vehicle symmetric cryptography," in *2011 IEEE Vehicular Technology Conference (VTC Fall)*, 2011, pp. 1–5.

[57] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth - a simple, backward compatible broadcast authentication protocol for CAN bus," in *ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, 2011, p. 20.

[58] Q. Wang and S. Sawhney, "VeCure: A practical security framework to protect the CAN bus of vehicles," in *2014 International Conference on the Internet of Things (IOT)*, 2014, pp. 13–18.

[59] J. Schmandt, A. T. Sherman, and N. Banerjee, "Mini-MAC: Raising the bar for vehicular security with a lightweight message authentication protocol," *Vehicular Communications*, vol. 9, pp. 188–196, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209616301619

[60] Z. Lu, Q. Wang, X. Chen, G. Qu, Y. Lyu, and Z. Liu, "LEAP: A lightweight encryption and authentication protocol for in-vehicle communications," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1158–1164.

[61] "NIST SP 800-38b – Recommendation for block cipher modes of operation: The CMAC mode for authentication," National Institute of Standards and Technology, Special Publication, 2005. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-38B

[62] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle CAN," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, 2015.

[63] S. Gürgens and D. Zelle, "A hardware based solution for freshness of secure onboard communication in vehicles," in *Computer Security*, S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, A. Antón, S. Gritzalis, J. Mylopoulos, and C. Kalloniatis, Eds. Cham: Springer International Publishing, 2019, pp. 53–68.

[64] A.-I. Radu and F. D. Garcia, "LeiA: A lightweight authentication protocol for CAN," in *Computer Security – ESORICS 2016*, I. Askoxylakis, S. Ioannidis, S. Katsikas, and C. Meadows, Eds. Cham: Springer International Publishing, 2016, pp. 283–300.

[65] ETSI, "Basic set of applications; definitions," ETSI, Technical Report on Vehicular Communications TR 102 638 V1.1.1, 2009.

[66] ——, "ITS-G5 access layer specification for intelligent transport systems operating in the 5 GHz frequency band," ETSI, European Standard EN 302 663 V1.3.1, 2020.

[67] T. Yoshizawa and B. Preneel, "Survey of security aspect of v2x standards and related issues," in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2019, pp. 1–5.

[68] ETSI, "ITS communications security architecture and security management," ETSI, Technical Specifications: Intelligent Transport Systems Security TS 102 940 v1.3.1, 2018.

[69] A. Bhargava, S. Verma, and B. K. Chaurasia, "Kalman filter for trust estimation in VANETs," in *2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON)*, 2015, pp. 1–6.

[70] N. Bißmeyer, S. Mauthofer, K. M. Bayarou, and F. Kargl, "Assessment of node trustworthiness in VANETs using data plausibility checks with particle filters," in *Vehicular Networking Conference (VNC)*. Seoul, South Korea: IEEE, 2012, pp. 78–85.

[71] S. Buchegger and J.-Y. Le Boudec, "Performance analysis of the confidant protocol," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, ser. MobiHoc '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 226–236. [Online]. Available: https://doi.org/10.1145/513800.513828

[72] C. A. Kerrache, C. T. Calafate, N. Lagraa, J. Cano, and P. Manzoni, "Hierarchical adaptive trust establishment solution for vehicular networks," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2016, pp. 1–6.

[73] R. Xing, Z. Su, N. Zhang, Y. Peng, H. Pu, and J. Luo, "Trust-evaluation-based intrusion detection and reinforcement learning in autonomous driving," *IEEE Network*, vol. 33, no. 5, pp. 54–60, 2019.

[74] C. A. Kerrache, N. Lagraa, C. T. Calafate, J.-C. Cano, and P. Manzoni, "T-VNets: A novel trust architecture for vehicular networks using the standardized messaging services of ETSI ITS," *Computer Communications*, vol. 93, pp. 68–83, 2016, multi-radio, Multi-technology, Multi-system Vehicular Communications. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366416302213

[75] Z. Yang, K. Yang, L. Lei, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, p. 11, 2019.

[76] W. Li and H. Song, "ART: An attack-resistant trust management scheme for securing vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 960–969, 2016.

[77] C. Zhang, K. Chen, X. Zeng, and X. Xue, "Misbehavior detection based on support vector machine and dempster-shafer theory of evidence in vanets," *IEEE Access*, vol. 6, pp. 59 860–59 870, 2018.

[78] S. A. Soleymani, A. H. Abdullah, M. Zareei, M. H. Anisi, C. Vargas-Rosales, M. K. Khan, and S. Goudarzi, "A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing," *IEEE Access*, vol. 5, pp. 15 619–15 629, 2017.

[79] R. G. Engoulou, M. Bellaiche, T. Halabi, and S. Pierre, "A decentralized reputation management system for securing the internet of vehicles," in *International Conference on Computing, Networking and Communications (ICNC)*. Honolulu, HI: IEEE, 2019, pp. 900–904.

[80] S. Sharma and A. Kaul, "A survey on intrusion detection systems and honeypot based proactive security mechanisms in vanets and vanet cloud," *Vehicular Communications*, vol. 12, pp. 138–164, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209617302784

[81] T. Rosenstatter and T. Olovsson, "Open problems when mapping automotive security levels to system requirements," in *Proceedings of the 4th International*

Conference on Vehicle Technology and Intelligent Transport Systems - Volume 1: VEHITS. SciTePress, Mar 2018, pp. 251–260.

[82] G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner, "SAHARA: A security-aware hazard and risk analysis method," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*. EDAA, 2015.

[83] T. Rosenstatter and T. Olovsson, "Towards a standardized mapping from automotive security levels to security mechanisms," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2018, pp. 1501–1507.

[84] T. Rosenstatter, K. Strandberg, R. Jolak, R. Scandariato, and T. Olovsson, "REMIND: A framework for the resilient design of automotive systems," in *2020 IEEE Secure Development (SecDev)*, Atlanta, GA, USA, 2020, pp. 81–95.

[85] V. Chang, M. Ramachandran, Y. Yao, Y.-H. Kuo, and C.-S. Li, "A resiliency framework for an enterprise cloud," *International Journal of Information Management*, vol. 36, no. 1, pp. 155 – 166, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S026840121500095X

[86] K. Strandberg, T. Rosenstatter, R. Jolak, N. Nowdehi, and T. Olovsson, "Resilient Shield: Reinforcing the resilience of vehicles against security threats," in *2021 IEEE 93th Vehicular Technology Conference (VTC2021-Spring)*, Helsinki, Finland, 2021, pp. 1–7.

[87] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/4/148

[88] T. Rosenstatter, C. Sandberg, and T. Olovsson, "Extending AUTOSAR's counter-based solution for freshness of authenticated messages in vehicles," in *IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2019, pp. 1–10.

[89] Q. Zou, W. K. Chan, K. C. Gui, Q. Chen, K. Scheibert, L. Heidt, and E. Seow, "The study of secure can communication for automotive applications," in *WCX 17: SAE World Congress Experience*. SAE International, mar 2017. [Online]. Available: https://doi.org/10.4271/2017-01-1658

[90] J. Van Bulck, J. T. Mühlberg, and F. Piessens, "VulCAN: Efficient component authentication and software isolation for automotive control networks," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC 2017. New York, NY, USA: ACM, 2017, pp. 225–237. [Online]. Available: http://doi.acm.org/10.1145/3134600.3134623

[91] M. Aramrattana, J. Detournay, C. Englund, V. Frimodig, O. U. Jansson, T. Larsson, W. Mostowski, V. D. Rodríguez, T. Rosenstatter, and G. Shahanoor, "Team Halmstad approach to cooperative driving in the grand cooperative driving challenge 2016," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1248–1261, April 2018.

[92] T. Rosenstatter and C. Englund, "Modelling the level of trust in a cooperative automated vehicle control system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1237–1247, April 2018.

[93] T. Rosenstatter, T. Olovsson, and M. Almgren, "V2C: A trust-based vehicle to cloud anomaly detection framework for automotive systems," in *The 16th International Conference on Availability, Reliability and Security*, ser. ARES 2021. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3465481.3465750

[94] P. Guo, H. Kim, L. Guan, M. Zhu, and P. Liu, "VCIDS: Collaborative intrusion detection of sensor and actuator attacks on connected vehicles," in *Security and Privacy in Communication Networks*, X. Lin, A. Ghorbani, K. Ren, S. Zhu, and A. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 377–396.

[95] T. Nandy, R. M. Noor, M. Yamani Idna Bin Idris, and S. Bhattacharyya, "T-BCIDS: Trust-based collaborative intrusion detection system for VANET," in *2020 National Conference on Emerging Trends on Sustainable Technology and Engineering Applications (NCETSTEA)*, Durgapur, India, 2020, pp. 1–5.

[96] E. A. Shams, A. Rizaner, and A. H. Ulusoy, "Trust aware support vector machine intrusion detection and prevention system in vehicular ad hoc networks," *Computers & Security*, vol. 78, pp. 245–254, 2018.

Chapter A

Open Problems when Mapping Automotive Security Levels to System Requirements

Adapted version that appeared in VEHITS 2018

T. Rosenstatter, T. Olovsson

Abstract. Securing the vehicle has become an important matter in the automotive industry. The communication of vehicles increases, they communicate with each other and to the infrastructure, they will be remotely diagnosed and provide the users with third-party applications. Given these areas of application, it is evident that a security standard for the automotive domain that considers security from the beginning of the development phase to the operational and maintenance phases is needed. Proposed security models in the automotive domain describe how to derive different security levels that indicate the demand on security, but do not further provide methods that map these levels to predefined system requirements nor security mechanisms. We continue at this point and describe open problems that need to be addressed in a prospective security framework for the automotive domain. Based on a study of several safety and security standards from other areas as well as suggested automotive security models, we propose an appropriate representation of security levels which is similar to, and will work in parallel with traditional safety, and a method to perform the mapping to a set of predefined system requirements, design rules and security mechanisms.

Open Problems when Mapping Automotive Security Levels to System Requirements

1 Introduction

New technologies and functionalities are constantly introduced to vehicles. Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication enables vehicles to share information with each other and send warnings, e. g., about roadworks and traffic jams. Remote diagnostics is performed by vehicle manufacturers and licensed repair shops, and platforms for third-party applications in vehicles are also provided. As a consequence of this ongoing transition, a security standard is crucial in order to secure vehicles against modifications, hacks, and espionage.

The exposure of serious vulnerabilities underlines the need for security in the automotive domain. The attack surface of modern vehicles was analysed by [Checkoway et al., 2011] and they demonstrated vulnerabilities in the Tire-Pressure Monitoring System, media-player, OBD-II port, and Bluetooth. Furthermore, [Miller and Valasek, 2014] provide a survey of attack surfaces of several vehicle models and [Yan, 2015] presents vulnerabilities in connected vehicles using different attack vectors.

A systematic approach dealing with the aforementioned security threats is necessary, and in addition, an automotive security standard is needed to harmonise the hardware and software security requirements between the vehicle manufacturers and the suppliers. This becomes evident when vehicle manufacturers get more dependent on their suppliers since the complexity of third-party modules is increasing and the modules are required to be reliable and secure. A security framework which also contains a mapping to system requirements and design rules containing guidelines describing how these demands can be fulfilled, increases the efficiency during the development and testing due to the fixed structure and guidance, as well.

The need for such a security standard has also been identified by the industry, which initiated the ISO/SAE AWI 21434 *Road Vehicles – Cybersecurity engineering*. This work item is ongoing and currently under development. Proposed security models, such as EVITA [Henniger et al., 2009] and HEAVENS [Islam et al., 2016], describe methods from identifying threats to classifying them into security levels. Both models focus on the identification of risks and threats, and how to classify them. HEAVENS describes methods to derive application specific requirements, but does not perform a mapping to predefined requirements nor security mechanisms that are required for each security level. EVITA on the other hand lists the results of their risk analysis in [Ruddle et al., 2009], but does not provide a mapping to generic security requirements based on the security level.

Information security is concerned with confidentiality, integrity and availability, which will be later extended according to STRIDE [ISO 15408, 2009, Microsoft

Corporation, 2005]. We define security levels similar to [Islam et al., 2016]. Security levels represent the necessity and extent for security measures in a specific function or module. The factors taken into account when deriving the security levels depend on the underlying security model that considers the severity of potential attacks, required expertise to perform the attack, or the impact for the involved parties in case of the attack. We split security requirements in two groups, system requirements and application specific requirements. System requirements are generic security requirements that need to be fulfilled for a certain security level and describe design rules or security functions. Application specific requirements are the result of a threat analysis and include individual requirements that are not covered by system requirements. Security mechanisms are methods to fulfil a requirement, for instance the choice of encryption algorithms to provide confidentiality of information, or use of access control lists to restrict the data flow in the in-vehicle network.

In this paper, we survey proposed security models and acclaimed standards in the area of safety and security, we investigate how these standards or models classify safety and security, and how they perform the transition to system requirements.

Our contributions in this paper are the following:

- A study on safety and security standards, along with proposed security models for the automotive domain.
- Propose methods for how to move forward from unique requirements of individual systems and identified security levels to a set of mandatory system requirements, design rules and security mechanisms and
- show that such requirements should be based on the security level of the function to be implemented.
- We show the benefits with having such a framework in place when dealing with third-party developed functionality.
- We show the challenges and complexity of defining such a framework.

2 Background and Related Work

The difference between safety and security is that safety is about handling malfunctioning behaviour that is caused by random errors. Security threats are caused by an attacker who intentionally wants to modify the system, harm involved people, or gather information. It may be also the case that the owner, who has physical access and unlimited time, slips in the role of an attacker in order to perform unauthorised modifications on the vehicle. The skills of attackers may vary from limited to advanced depending on his knowledge, purpose, and equipment. For these reasons, security involves complex countermeasures. An attacker who has successfully exploited a vulnerability of one vehicle is able to apply the same method to all vehicles of that specific model or even to all vehicles of that manufacturer or supplier depending on the kind of vulnerability [SAE J3061, 2016].

The large attack surface in vehicular security is another difference to safety. Performing Hazard Analysis and Risk Assessment (HARA) on functional safety requires a narrower focus compared to assessing the security of a system. The security assessment of a system requires one to cope with a larger attack surface. For instance, a vehicle that is able to communicate with its infrastructure and cooperate with other vehicles can not only be attacked locally, it can be attacked through this communication channel as well. Moreover, vehicles have Internet access which additionally increases the attack surface. Having a built-in mobile communication unit also enables attacks via SMS and other phone services. In addition to the wireless communication, Checkoway et al. showed other attack vectors, such as an attack through the media-player [ENISA, 2016, Checkoway et al., 2011].

Looking at the different areas shows that safety has been adapted to the specific needs for this area. [ISO 26262, 2011] is the standard for functional safety of road vehicles, [RTCA DO-178, 2011] and [RCTA DO-254, 2000] (software and hardware) are customised for the aeronautics domain, and the railway domain is described in CENELEC EN 50126, CENELEC EN 50128, and CENELEC EN 50128. [Blanquart et al., 2012] perform a comparison of the criticality categories across safety standards in different domains, including the aforementioned standards and the corresponding safety standards for nuclear facilities and space systems. They highlight how these domains differ from each other in terms of structure and guidance throughout the development process.

Security standards exist in many areas, such as programming, industrial automation, and system and device security. The [SEI CERT C, 2016] Coding Standard, for instance, defines rules for specific programming languages, in this case for C, and uses a classification in levels to indicate the impact of not addressing a certain rule. SEI CERT C shows how certain programming traits have to be implemented in order to be reliable, secure and safe. By following these rules, undefined behaviour that may lead to vulnerabilities will to a large extent be eliminated. The standard ranks each rule with an example and its priority and level. A combination of severity, likelihood and remediation costs results in such a level ranging from 1 to 3. The priorities are directly mapped to the levels.

[NIST SP 800-53r4, 2013] is a catalogue of security controls and assessment procedures for information systems. The security controls are split in 18 families, such as Access Control, Incident Response, and Identification and Authentication. Consequently, each family comprises security controls mapped to priority and impact levels (*low*, *medium*, *high*). Security controls are nested, the lowest priority level has to be implemented first and controls with higher priority have to be implemented in addition to the controls with a lower priority level.

Security models for the automotive domain have been proposed by various researchers. [Burton et al., 2012] suggest a method that extends ISO 26262 with security analysis. A combined safety and security development lifecycle is presented by Schmittner et al. in [Schmittner and Ma, 2014] and [Schmittner et al., 2015]. Burton et al. and Schmittner et al. both focus on the differences between safety and security and how to combine them, but they do not discuss how predefined system requirements or design rules for security can be defined. Common Criteria [ISO 15408, 2009] is a standard for evaluating security properties of systems and devices.

The similarities of ISO 26262 and Common Criteria are discussed in [Schmittner and Ma, 2014]. The authors describe the relationship between the Automotive Safety Integrity Levels (ASILs) from ISO 26262 and the Evaluation Assurance Levels (EALs) from Common Criteria according to the strictness and degree of formalism. The SAHARA method presented by [Macher et al., 2015] combines the existing HARA known from ISO 26262 with a security assessing method considering the needed resources and know-how. Macher et al. do not describe a method to derive system requirements based on the resulting security level. The EVITA [Henniger et al., 2009] and HEAVENS [Islam et al., 2016] models describe the procedure on how to derive security levels and how these can be used for requirement engineering, however, they do not perform a mapping to system requirements as we propose.

Guidelines for cybersecurity with respect to the automotive domain are [SAE J3061, 2016] and [ENISA, 2016]. Both guidelines provide good practice examples and recommendations, whereas ENISA limits the scope by excluding autonomous vehicles and V2V communication in their guideline. Another difference is that J3061 sets the focus on the necessary processes and their implementation, while ENISA describes the typical architecture of smart vehicles and possible threats and attacks. J3061 lists Threat, Vulnerabilities, and implementation Risks Analysis (TVRA), which is a threat and risk assessment method developed by ETSI in TS 102 165-1 [ETSI, TS, 2011]. According to J3061, this model is not suited for control and data networks of vehicles, as it was developed specifically for telecommunications networks. These two guidelines for cybersecurity address important subjects, but do not discuss methods for mapping to system requirements nor mechanisms.

3 The Complexity in Automotive Security

Lifetime. The automotive domain differs in many ways from other areas. A vehicle has a lifetime of about 150.000 to 300.000 km [Hawkins et al., 2012]. During this time, the vehicle has to be safe, secure, firmware needs to be updated, the owner may change, and vehicle parts or modules need to be replaced. Discovering a security vulnerability in the vehicle requires a fast reaction and update distribution to the vehicles. Once a severe security problem has been identified, over-the-air updates provide efficient means for distribution as they are much faster than a recall of a certain vehicle model or vehicles with a specific component from a supplier. In addition, over-the-air updates are needed because security requirements 20 years from now are likely to be different from what is designed today, since the expected lifetime from design of security functionality to the expected end of the vehicle lifetime can be as much as 20 – 25 years.

Interplay between safety and security. The safety of the passengers has to be retained in all situations. Fault detection mechanisms have to be designed in such a way that they cannot be exploited by an attacker. For example, the use of redundant modules for increased safety, may open up for attacks where both modules believe the other one is active while it is the attacker who sends the messages.

Compliance to standards. One specific challenge for heavy duty vehicles is the required compliance to [SAE J1939, 2013]. This standard specifies the exact content of frames that have to be transmitted within the in-vehicle network. To comply, the frames may not be changed and thus encryption mechanisms may not be used. This restriction limits the set of suitable security mechanisms.

Compliance between manufacturers and suppliers. The suppliers will provide modules with more functionalities and may also need to maintain the security of their products. Manufacturers integrate software, and hardware modules from third-party developers and in-house developed modules into a vehicle and thus need to ensure the security and safety of all modules individually and combined. A well-defined framework with strict system requirements for security functions and mechanisms to be used, would simplify the requirement specifications and communications between these two parties.

Maintenance. Authorised workshops need to be able to diagnose the vehicle and replace modules in case of a failure. For this reason, they need to, for instance, be able to handle the change of security keys in an offline and online environment. Authorised devices may also need to be revoked in case of theft.

Alignment with ISO 26262. The harmonisation/alignment with the functional safety standard for road vehicles [ISO 26262, 2011] is important when introducing a new framework for automotive security. ISO 26262 has a high acceptance in the automotive domain and would, for this reason, significantly reduce the time required for introducing such a security framework, because of the already known processes. The question is to which extent it should be harmonised. Strong harmonisation has the advantage of easier integration due to known processes, but it may not be ideal for security due to the fundamental differences to safety.

Guidance. The necessary level of guidance has to be defined. Providing a strict guidance for each requirement may not be feasible or optimal for certain cases. The developers may, for certain security requirements, find other countermeasures that are more suitable. On the other hand, sparse guidance leads to overhead when evaluating security of third-party components and individual solutions may not correspond to best practices.

We address these problems by investigating how standards and other security models handle guidance and propose a method able to cope with the problems listed above.

4 Standards and Models

In this section, we describe standards, models, and approaches that influence the design of safety and security by assigning levels according to a defined scheme. They are all well-known and accepted in their domain or applicable for the automotive area. The insights into how these standards allocate safety or security levels and how they perform the mapping to requirements are further used for our suggested framework. The related standards and models being described in detail are [ISO 26262, 2011], [RTCA DO-178, 2011], EVITA [Henniger et al., 2009], HEAVENS [Islam

et al., 2016], Trust Assurance Levels [Kiening et al., 2013], and [IEC 62443, 2013].

First, we describe the purpose of the standards and models followed by a discussion on the number of levels their analysis results in. Next, we discuss the impact of the levels in the design and development of the system and investigate how the standards or models address the relation between requirements and allocated levels. Do they provide strict information about the requirements necessary for each level or is the mapping between the allocated levels and the requirements without guidance?

4.1 Safety Standards

ISO 26262 applies HARA for a system without safety measures by taking the severity, exposure, and the controllability into account. A hazard is defined by ISO 26262 as malfunctioning behaviour that potentially causes harm. After identifying the levels for each class, such as severity and controllability, they are mapped according to a predefined matrix to the ASIL levels. The ASIL levels comprise QM, A, B, C, and D, where QM corresponds to *Quality Management*, i.e., a non-safety relevant event which does not require any further safety consideration in the design and development of the system. Events classified as ASIL D, the highest level, require the highest demand regarding risk reduction. In case a system failure causes several hazards, the highest occurring ASIL rating has to be applied [ISO 26262, 2011].

Next, a safety goal is defined for each hazardous event. The safety goals are subsequently associated with a functional safety concept, which states how the safety goal can be achieved. The next step is to formulate a technical safety concept describing how the functionality is going to be implemented by hardware and software on the system level. The software and hardware safety requirements describe the specific requirements, which will be implemented. The requirements inherit their ASIL level from their safety goal respectively their parent requirement [ISO 26262, 2011].

ISO 26262 provides guidelines and requirements for each level on system, hardware, software, production, and operation level. In addition to the specified requirements for each ASIL level, this standard also provides three levels of recommendations, no recommendation (○), recommended (+), and highly recommended (++). The method *independent parallel redundancy* as a mechanism for error handling at the software architectural level is recommended for ASIL C and highly recommended for ASIL D [ISO 26262, 2011].

DO-178 is the safety standard for the aeronautics domain, its categories are called Development Assurance Levels (DAL). The DALs are representing the effects of a failure condition, e. g., catastrophic or hazardous. The five DALs range from A, the most demanding level, to E, which is the equivalent to ASIL QM. A top-level function is mapped to the Function DAL (FDAL) according to a table that associates the failure condition class and the quantitative safety requirement (failures per hour) with the DAL. These top-level functions are decomposed to sub-functions, which are further decomposed to items. It may be the case that a top-level function is divided into more than one sub-function, where one of the sub-functions has a lower or the same DAL as the top-level function. DO-178 also provides guidance, but not to the same extent as ISO 26262 [Blanquart et al., 2012, RTCA DO-178, 2011, RCTA DO-254, 2000].

4.2 Security Models in the automotive domain

The HEAVENS and EVITA projects define models to derive security levels. [Henniger et al., 2009] describe a model developed in the EVITA project. [Islam et al., 2016] propose an ISO 26262 compliant model as part of the HEAVENS project. Both models specify how to identify threats and classify them into security levels.

Henniger et al. use attack trees based on use cases as base for the following requirements analysis. The root of the attack tree is the goal of the attack. The sub-levels contain sub-goals that can lead to the goal of the parent node.

A risk assessment is performed for each potential attack. The mapping of the three components, severity (vector), probability of a successful attack, and controllability, is derived from a predefined table which leads to a security risk level. This level is not associated to a single value, it is a 4-component vector describing the security risk level for the elements of the severity vector. The levels for each element are in the range $[0, 7]$, where 0 represents no risk and 6 the highest risk. Level 7 and 7+ are used for safety-critical threats with controllability $C \geq 3$ and severity *high* [Henniger et al., 2009].

The authors further discuss how the security risk levels can be used to prioritise security requirements. Requirements resulting from the developed use cases and risk assessment are listed in [Ruddle et al., 2009]. Henniger et al. highlight that not only the highest rating should be considered, the number of occurrences in the attack trees has to be considered as well. A lower risk that is seen in several attack trees may have the same importance as a level 5 or higher risk that appears only once in the attack tree [Henniger et al., 2009].

The HEAVENS risk assessment model suggested by Islam et al. describes the workflow for identifying assets and threats (threat analysis), and a method describing how to perform the risk assessment. Islam et al. apply Microsoft's STRIDE model [Microsoft Corporation, 2005] to identify the asset/threat pairs.

The result of the risk assessment is the security level, which is a combination of the threat level and the impact level. The levels for the resulting security level are QM, low, $medium$, $high$, $critical$. Islam et al. highlight the parallels to ISO 26262. The functional safety requirements derived from the safety goals in ISO 26262 have the same property as the high-level security requirements originating in the asset/threat pair and their corresponding security level. Both are high-level requirements that are independent from the implementation. These requirements are consequently divided into technical security requirements on system level, which further result in hardware and software security requirements.

4.3 Trust Assurance Levels for V2X communication

The purpose of the Trust Assurance Levels is to classify the security of Vehicle-to-Everything (V2X) communication nodes. [Kiening et al., 2013] provide the minimum requirements for each Trust Assurance Level (TAL) and discuss the benefits of certifying V2X nodes and how to perform the verification of security. The mapping of the TAL is performed according to a predefined table. This table contains the minimum requirements and a description of security implications for each level.

The proposed levels are nested and range from 0 to 4. A node with TAL 0 does not have any security measures. With an increasing TAL, the core V2X communication modules and other relevant modules of the node need to be secured. For instance, TAL 4 requires all involved modules to be protected. Moreover, the authors map the TALs to the EALs of Common Criteria [Kiening et al., 2013].

4.4 Cyber Security Standard - IEC 62443

[IEC 62443, 2013] is a group of security standards for Industrial Automation and Control Systems. Part 3 describes the system security requirements and security levels. The security levels range from 0 to 4 and are further split into Target Security Levels (SL-T), Achieved SLs (SL-A), and Capability SLs (SL-C). A security level of 0 corresponds to no specific requirements for security and level 4 implies the highest demand on security. SL-T is derived from a consequence analysis of a particular system, called zone, and describes the desired security level. During the iterative design phase SL-A and SL-T are compared with each other after every cycle. Components and systems need to provide the SL-C that indicates its capability in regards to the defined security levels. In case that SL-C does not meet the required SL-T, compensating countermeasures have to be implemented [IEC 62443, 2013].

The high-level requirements are named Foundational Requirements (FRs) and consist of seven elements, e. g., system integrity, data confidentiality, and use control. These FRs are consequently broken down into System Requirements (SRs) and Requirement Enhancements (REs). The security level for a specific zone or component does not consist of a single value, it is a 7-component vector describing the security level for each FR. A table in [IEC 62443, 2013] maps the SRs and REs to the security level of each FR.

Equation A.1 shows the composition of the Security Levels (SLs) in IEC 62443. Each component listed in this vector refers to a FR. An example shown in IEC 62443-3-3 is the SL-T of a basic process control system zone. It is specified that this zone requires a SL of 3 for the FRs *Restricted data flow* and *Resource availability*. In contrast, measures to provide *Data confidentiality* are not required, as the SL is 0.

Table A.1 provides an overview of the reviewed standards and models. It highlights the differences between safety and security. Functional safety standards have five different levels whereas the automotive security models, such as EVITA and HEAVENS, use a more complex representation of security or risk levels. IEC 62443, developed for the security of industrial automation and control systems, classifies security as a 7-component vector with a range of five levels for each element and differs compared to ISO 26262 with respect to how it relates the SLs to requirements.

$$SL = \begin{bmatrix} \text{Identification \& authentication ctrl.} \\ \text{Use control} \\ \text{System integrity} \\ \text{Data confidentiality} \\ \text{Restricted data flow} \\ \text{Timely response to events} \\ \text{Resource availability} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 0 \\ 1 \\ 3 \\ 1 \\ 3 \end{bmatrix} \tag{A.1}$$

Table A.1: Overview of the reviewed standards in respect to their classification approach.

Standard/Framework	Area	# Levels	Vector size	Predef. SRs
ISO 26262	Safety	5	1	(✓)[a]
DO-178	Safety	5	1	–
IEC 62443	Security	5	7	✓
TAL	Security	5	1	✓
EVITA	Security	8	4	–
HEAVENS	Security	5	1[b]	–

[a] ISO 26262 provides recommendations for specific methods depending on the SL.
[b] HEAVENS associates each threat/asset pair with a SL.

5 Proposed Security Levels and Mapping

Our previous discussion about the problems of implementing security in vehicular systems followed by a survey of standards and models, are the base for the following suggestion for the number and representation of security levels, and the mapping to system requirements, design rules and security mechanisms. Investigating how established standards and proposed security models define a classification in form of Security Levels (SLs) is important for suggesting an automotive security framework. Additionally, we discuss how to evaluate or even certify the security compliance of a module, and how well our proposed framework is aligned with ISO 26262. We want to highlight that the suggested solutions are a proposal based on the review of several standards and models from safety or other security domains and should provide a recommendation to future research.

5.1 Number of Security Levels

The decision on the number of SLs and the mapping to system requirements strongly depends on the underlying model for risk/threat assessment. Models differ in their composition and weights for parameters, such as expertise to perform the attack, opportunity, and impact level.

Both functional safety standards, ISO 26262 and DO-178, use five levels to classify safety, but security models propose different solutions. The TALs with the focus on V2X communication security use five levels. Two automotive security models propose a more complex classification of security or risk levels. EVITA focusses on the risk of security relevant attacks by representing the risk as a 4-component vector and HEAVENS associates the SLs with a specific asset/threat pair.

Following the HEAVENS approach by using its classification of SLs leads to one

level in the range of 0 to 4 for each threat/asset pair, meaning that the threat violating confidentiality of an individual asset ranges between these levels. Continuing with defining high-level requirements and technical security requirements for each threat/asset pair would only result in more overhead. Instead, we recommend the use of system requirements, which describe the necessary security measures for each type of threat and security attribute and thus provide the developer already with necessary requirements that need to be fulfilled depending on the SL.

The classification of SLs needs to provide sufficient categories to have a distinct separation of the required security measures. Having a wide range of SLs may lead to an overly detailed guidance, which may be inefficient due to the high granularity of requirements and the difficulty to distinguish between the SLs.

Since security needs to address many different aspects or attributes, such as the authenticity of messages and their origin, system integrity, and data confidentiality, it is reasonable to use a vector defining the SL for each component representing one of these areas. The SL of each component is in the range of 0 to 4. Such an approach is described in IEC 62443 and seems to be appropriate for the automotive domain as well, as the representation as a single value may lead to imbalanced security measures. For certain modules or subsystems, it might be necessary to provide data confidentiality, whereas data integrity might be of greatest importance for many other subsystems, hence, it is benefiting to distinguish between such attributes and assign them SLs individually. In addition, a vector representation eases the communication between the parties, as a vector already combines the demanded level of security for each attribute.

Furthermore, we suggest to distinguish between target, achieved and capability SL, as described in IEC 61442. Modules provided by suppliers, or in-house developed modules should be classified according to their security capability (SL-C). This may lead to a more efficient reuse of developed modules, as they are clearly marked with the SL they are capable of. This approach also simplifies the design of the system, since modules and systems can be labelled with their target SL (SL-T) that states the necessary system requirements and design rules.

Describing security as a vector instead of a single value is different to ISO 26262, nevertheless, we believe that it is unavoidable to present security as a structure describing several attributes. As an example, $SL(auth.) = 1$ may require a verification of the new firmware when performing an upgrade, whereas $SL(auth.) = 3$ may require a firmware verification at every start-up and $SL(auth.) = 4$ may additionally require the authenticity of messages sent and received within this particular subsystem. IEC 62443 and other security models also use a vector or other similar approaches and thus support our choice.

5.2 Mapping to Security Requirements and Mechanisms

There are different ways to map SLs to system requirements and security mechanisms. One option is to perform a binary mapping of the SLs, e. g., a system requirement has to or does not have to be fulfilled. An alternative is the introduction of recommendations in combination to the binary mapping, which is a closer approach to ISO 26262. The presentation of the requirements for certain SLs, is another important aspect.

Table A.2: Binary mapping of security levels to system requirements and requirement enhancements.

FR 1	0	1	2	3	4
	none	*low*	*medium*	*high*	*critical*
SR 1		•	•	•	•
SR 2			•	•	•
RE 2.1				•	•
RE 2.2					•
SR 3			•	•	•
…					

ISO 26262 lists the requirements and recommendations in separate documents, e. g., description for system, hardware and software level. IEC 62443 on the other hand provides a compact overview of the required security measures. The System Requirements (SRs) and Requirement Enhancements (REs) of a Foundational Requirement (FR) are mapped to the SLs. The demands for fulfilling a SL of a FR is reflected in the required SRs and REs. For lower levels, it is sufficient to only fulfil a few requirements, but in order to provide the highest SL, one must fulfil all SRs and REs. This way, it is ensured that also modules or subsystems with lower SLs provide basic methods to ensure a specific security attribute. Table A.2 illustrates the structure of this approach. It shows which SRs and REs of FR 1 are required for each SL. For instance, SL 1 requires only SR 1, whereas SL 2 requires SR 1, SR 2, and SR 3. Higher levels demand also the RE 2.1 respectively RE 2.1 and RE 2.2.

The FRs and their associated SRs and REs shown in Table A.2 still need to be defined. One approach is to use the FRs of IEC 62443 (see Equation A.1) as a base and adjust the SRs and REs by incorporating automotive specific requirements which are described in Section 3, e. g., the possibility to exchange vehicle parts in an offline environment (offline distribution of cryptographic keys) and offline diagnostics in a workshop (see Section 3). Another approach we propose is to combine the structure of IEC 62443 with HEAVENS – the FRs are the security attributes, which are mapped to Microsoft's STRIDE model [Islam et al., 2016, Microsoft Corporation, 2005]. This leads to a 6-component vector for each asset, as shown in Equation A.2. This example shows the security demands for each element, e. g., the demands for integrity are *high*, whereas there are no demands on confidentiality. Additionally, SRs and REs for each FR have to be defined and mapped to the SLs.

With this approach, it is possible to incorporate design rules in the SRs as well. Plausible design rules may be the physical isolation of critical networks, multi-factor authentication of diagnostic devices or other modules with a critical SL, or the composition of modules with different SLs in one control unit.

$$SL = \begin{bmatrix} \text{Authenticity} \\ \text{Integrity} \\ \text{Non-repudiation} \\ \text{Confidentiality} \\ \text{Availability} \\ \text{Authorisation} \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 0 \\ 2 \\ 1 \end{bmatrix} \tag{A.2}$$

A mapping to specific security mechanisms can be performed through recommendations. The challenge when introducing such a mapping is its dynamics. The mapping changes over time as cryptographic algorithms may be considered as broken or existing hardware may have sufficient processing capabilities to solve the cryptographic problem on which the mechanism is built upon. Such recommendations support the developers in choosing mechanisms that satisfy a certain SL. They can be represented as a list of requirements associated with certain mechanisms. This method requires an individual identifier or a version number for each mapping to a security mechanism in order to provide a seamless documentation of how the SRs have been addressed.

Tools for the secure implementation of software are the [SEI CERT C, 2016] Coding Standard and the [MISRA C:2012, 2013] Guideline, both give guidance and provide rules to develop software that is safe, secure, and reliable. ISO 26262, for instance, recommends the use of MISRA C. We propose to comply with such a secure coding standard for any SL greater than 0, as basic vulnerabilities inherited from programming languages or the wrong use of it can be limited this way.

We believe that a vector representation allows the combination of demands from different disciplines, such as safety. Adding the ASIL levels from ISO 26262 to the vector is beneficial when discussing and deploying the requirements for a module or subsystem, as the requirements of both areas have to be implemented in the very same module or feature. Providing a vector or table as illustrated in Table A.3 is thus useful for the software architects and developers to see if required safety mechanisms interfere with security requirements or vice versa. We propose such a combined presentation of both, safety and security demands, as it is necessary in order to fulfil the necessary requirements.

5.3 Evaluation and Certification

Providing evidence about how the SL has been achieved is necessary for the vehicle manufacturers in order to rely on the security capabilities of modules offered by suppliers. Common Criteria, a standard for IT security evaluation, specifies methods and requirements for each Evaluation Assurance Level. Schmittner et al. perform in [Schmittner and Ma, 2014] a mapping of the levels defined in Common Criteria and the ASIL levels from ISO 26262. [Wooderson and Ward, 2017] describe how the assessment as in Common Criteria can be applied for cybersecurity in vehicular systems. They further discuss the benefits and disadvantages of an internal assessment and an independent certification body.

Table A.3: Combined presentation of security and functional safety levels.

		0	1	2	3	4
		QM	*A*	*B*	*C*	*D*
Security	Authenticity			•		
	Integrity				•	
	Non-repudiation		•			
	Confidentiality	•				
	Availability			•		
	Authorisation		•			
Safety				•		

We suggest, that evidence for compliance in form of documents, such as the attack tree analysis or the threat analysis and an overview how the identified threats have been addressed, is sufficient for lower SLs. However, components requiring a SL of high (3) or critical (4) need to provide a more detailed documentation on how the security measures are taken into account.

As the SLs consist of a vector, it is possible to define the level of detail for the required documentation for each element (FR) of the vector. This way, it is ensured that no unnecessary overhead for documentation has to be performed.

6 Conclusion

With the increasing functionality of modern vehicles, it is essential to have a standardised security framework for vehicles that specifies the development lifecycle as well as System Requirements (SRs). A standard, such as ISO 26262 for functional safety of road vehicles, is needed so that all involved parties, e. g., manufacturer and suppliers, share the same understanding for automotive security.

We provide a study on several safety and security standards from different domains and discuss specific problems that have to be solved before a similar security standard can be introduced in the automotive domain. Based on this, we suggest a representation of Security Levels (SLs), how to map them to SRs, and discuss how the security compliance of systems and modules with a certain SL can be proven.

The number of SLs found to be suitable is five. Having five SLs not only harmonises with ISO 26262, it allows also a sufficient guidance through specifying system requirements and suitable security mechanisms. A higher number of levels leads to a stricter guidance and would only increase the complexity of the mapping to requirements. However, as shown in IEC 62443 and two automotive security models, security needs to address different attributes or categories, e. g., confidentiality and integrity. For this reason, we propose to define one SL for each category and conse-

quently use a vector for representation. Due to the use of a vector, we also suggest to include safety as one element in order to provide a matrix that presents all safety and security demands, as safety measures may interfere with security. Furthermore, we propose the use of a capability SL (SL-C), similar to IEC 62443, to be used as a classifier for the security of third-party modules and all in-house developed modules.

Providing guidance in the process of mapping SLs to system requirements as part of a security framework has benefits, such as a common understanding of the required security for each level, and the compliance of products from suppliers. Adding recommendations for how to implement certain SRs by providing suitable mechanisms, further guides the developer. It has to be highlighted that such recommendations need to be continuously maintained and updated as security mechanisms might have to be revised due to published exploitation methods.

The evaluation and compliance of components is important for the vehicle manufacturer. During the concept and design phase it has to be known what the system or subsystems need to be capable of and what components provided by suppliers are capable of. For lower SLs, we believe that the documentation of the threat analysis and attack tree analysis together with documented proof providing information how these security requirements have been handled, is sufficient. For the SL high and critical, we propose to adapt Common Criteria according to the specific needs in the automotive domain.

In future work, the detailed security requirements and mechanisms have to be identified, evaluated for their applicability in the automotive domain, and mapped to SLs. Additionally, it is necessary to include this proposed structure in a security framework that is suitable for this domain.

Acknowledgements. This research was funded by the HoliSec project (2015-06894) funded by VINNOVA, the Swedish Governmental Agency for Innovation Systems.

Bibliography

[Blanquart et al., 2012] Blanquart, J.-P., Astruc, J.-M., Baufreton, P., Boulanger, J.-L., Delseny, H., Gassino, J., Ladier, G., et al. (2012). Criticality categories across safety standards in different domains. *ERTS-2012, Toulouse*, pages 1–3.

[Burton et al., 2012] Burton, S., Likkei, J., Vembar, P., and Wolf, M. (2012). Automotive functional safety = safety + security. In *Proceedings of the First International Conference on Security of Internet of Things - SecurIT 12*. Association for Computing Machinery (ACM).

[Checkoway et al., 2011] Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., et al. (2011). Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*. San Francisco.

[ENISA, 2016] ENISA (2016). Cyber Security and Resilience of smart cars. Technical report, The European Union Agency for Network and Information Security (ENISA).

[ETSI, TS, 2011] ETSI, TS (2011). 102 165-1: Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN). *Methods and protocols*.

[Hawkins et al., 2012] Hawkins, T. R., Gausen, O. M., and Strømman, A. H. (2012). Environmental impacts of hybrid and electric vehicles—a review. *The International Journal of Life Cycle Assessment*, 17(8):997–1014.

[Henniger et al., 2009] Henniger, O., Apvrille, L., Fuchs, A., Roudier, Y., Ruddle, A., and Weylr, B. (2009). Security requirements for automotive on-board networks. In *9th International Conference on Intelligent Transport Systems Telecommunications, (ITST)*. Institute of Electrical and Electronics Engineers (IEEE).

[IEC 62443, 2013] IEC 62443 (2013). IEC 62443 – Industrial communication networks - Network and system security. Standard, International Electrotechnical Comission.

[Islam et al., 2016] Islam, M. M., Lautenbach, A., Sandberg, C., and Olovsson, T. (2016). A risk assessment framework for automotive embedded systems. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security - CPSS 16*. Association for Computing Machinery (ACM).

[ISO 15408, 2009] ISO 15408 (2009). ISO/IEC 15408:2009 Information technology – Security techniques – Evaluation criteria for IT security. Standard, International Organization for Standardization (ISO).

[ISO 26262, 2011] ISO 26262 (2011). ISO 26262:2011 Road Vehicles – Functional Safety. Standard, International Organization for Standardization (ISO).

[Kiening et al., 2013] Kiening, A., Angermeier, D., Seudie, H., Stodart, T., and Wolf, M. (2013). Trust assurance levels of cybercars in V2X communication. In *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles - CyCAR 13*. Association for Computing Machinery (ACM).

[Macher et al., 2015] Macher, G., Sporer, H., Berlach, R., Armengaud, E., and Kreiner, C. (2015). SAHARA: A security-aware hazard and risk analysis method. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*. EDAA.

[Microsoft Corporation, 2005] Microsoft Corporation (2005). The stride threat model. Available at https://msdn.microsoft.com/en-us/library/ee823878.aspx. (Accessed: 2017-11-06).

[Miller and Valasek, 2014] Miller, C. and Valasek, C. (2014). A survey of remote automotive attack surfaces. *Black Hat USA*.

[MISRA C:2012, 2013] MISRA C:2012 (2013). *MISRA C: Guidelines for the Use of the C Language in Critical Systems 2012*. Motor Industry Research Association.

[NIST SP 800-53r4, 2013] NIST SP 800-53r4 (2013). NIST Special Publication 800-53 – Security and Privacy Controls for Federal Information Systems and Organizations. Standard, National Institute of Standards and Technology.

[RCTA DO-254, 2000] RCTA DO-254 (2000). Design assurance guidance for airborne electronic hardware. Standard, RTCA and EUROCAE.

[RTCA DO-178, 2011] RTCA DO-178 (2011). Software considerations in airborne systems and equipment certification. Standard, RTCA and EUROCAE.

[Ruddle et al., 2009] Ruddle, A., Ward, D., Weyl, B., Idrees, S., Roudier, Y., Friedewald, M., Leimbach, T., et al. (2009). Deliverable D2.3: Security requirements for automotive on-board networks based on dark-side scenarios. Deliverable, E-safety vehicle intrusion protected applications (EVITA).

[SAE J1939, 2013] SAE J1939 (2013). Serial Control and Communications Heavy Duty Vehicle Network - Top Level Document. Technical report, SAE International.

[SAE J3061, 2016] SAE J3061 (2016). SAE J3061: SURFACE VEHICLE RECOMMENDED PRACTICE - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems. Standard, SAE International.

[Schmittner and Ma, 2014] Schmittner, C. and Ma, Z. (2014). Towards a framework for alignment between automotive safety and security standards. In *International Conference on Computer Safety, Reliability, and Security*, pages 133–143.

[Schmittner et al., 2015] Schmittner, C., Ma, Z., and Schoitsch, E. (2015). Combined safety and security development lifecylce. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. Institute of Electrical and Electronics Engineers (IEEE).

[SEI CERT C, 2016] SEI CERT C (2016). Sei cert c coding standard rules for developing safe, reliable, and secure systems. book, Carnegie Mellon University.

[Wooderson and Ward, 2017] Wooderson, P. and Ward, D. (2017). Cybersecurity testing and validation. In *SAE Technical Paper*. SAE International.

[Yan, 2015] Yan, W. (2015). A two-year survey on security challenges in automotive threat landscape. In *2015 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE.

Chapter B

Towards a Standardized Mapping from Automotive Security Levels to Security Mechanisms

Adapted version that appeared in ITS-C 2018

T. Rosenstatter, T. Olovsson

Abstract. Modern vehicles are becoming targets and need to be secured throughout their lifetime. There exist several risk assessment models which can be used to derive security levels that describe to what extent components, functions and messages (signals), need to be protected. These models provide methods to gather application-specific security requirements based on identified threat and item combinations that need to be coped with. However, a standardised mapping between security levels and required mandatory security mechanisms and design rules is currently missing. We address this problem first by suggesting that the risk assessment process should result in five security levels, similar to the functional safety standard ISO 26262. Second, we identify suitable security mechanisms and design rules for automotive system design and associate them with appropriate security levels. Our proposed methodology is as much as possible aligned with ISO 26262 and we believe that it should therefore be realistic to deploy in existing organisations.

Towards a Standardized Mapping from Automotive Security Levels to Security Mechanisms

1 Introduction

Computer and network security became more and more important as the number of interconnected devices spread. We now face similar challenges in the ongoing transition towards the Internet of things, industry 4.0 and smart connected vehicles. Constraints, such as low computational power, low energy consumption and real-time reaction, are major challenges that limit the range of applicable security mechanisms and design rules. Back in 2011, Checkoway et al. provided a detailed analysis of the attack surface of vehicles including attacks via the Tire Pressure Monitoring System and the media player [1]. Furthermore, Miller and Valasek have demonstrated several vulnerabilities that lead to attacks against vehicles that could be performed remotely in 2015 [2].

There is currently no standard similar to the functional safety standard ISO 26262 [3] for security in the automotive domain. The ISO work item ISO/SAE AWI 21434 *Road Vehicles – Cybersecurity engineering* is currently under development and is planned to address threat modeling and risk assessment. Proposed security models, such as Microsoft's STRIDE threat model [4], SAE J3061 [5] and the HEAVENS model [6], describe how to identify items that need to be secured, perform the threat analysis and result in security levels for components and functions in a vehicle. HEAVENS additionally describes a method for deriving item specific requirements. Currently missing in proposed models is how to select required security mechanisms for each component and function based on the security level when following the risk assessment process. In this paper, we address this problem by first suggesting that the risk assessment process should result in *five* security levels, similar to the Automotive Safety Integrity Levels (ASILs) and, when needed, a set of specific security requirements for that particular function. Figure B.1 provides an overview and puts our work into context.

We believe the proposed methodology can show the way towards a possible future automotive security standard similar to ISO 26262 used in automotive safety. The proposed methodology is realistic to deploy in existing organizations, as it is aligned with safety design methodologies which are already in place. In addition, having distinct requirements for security design allows the classification of components, including third-party designs, and enables us to know how robust and secure the design is and to what extend it can be trusted by other components.

We emphasize that we are focusing on mechanisms to be deployed in vehicles rather than traditional information systems, which is covered by the ISO 27000 family [7].

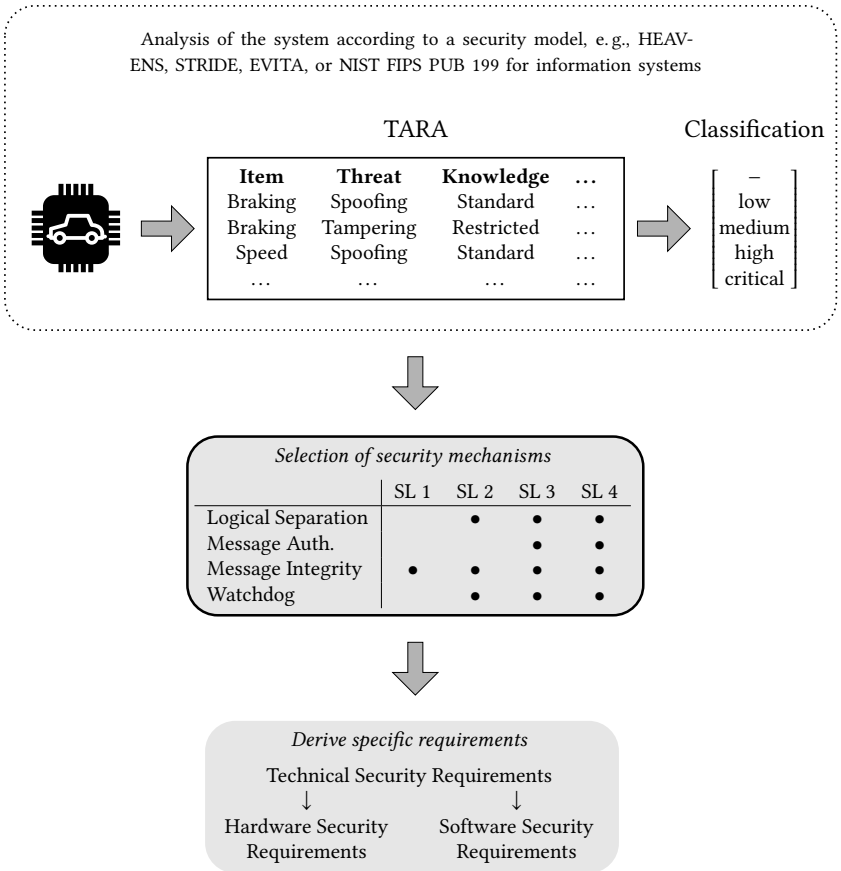


Figure B.1: Overview of the steps from performing Threat Analysis and Risk Assessment (TARA) to requirement engineering.

The contributions of this paper are as follows:

- We propose a set of suitable mandatory security mechanisms and design rules for the automotive domain.
- We suggest a framework for mandatory security mechanisms that should be required for specific security levels.
- We motivate the proposed method with an automotive use case.

2 Background and Related Work

The Cybersecurity Guidebook for Cyber-Physical Vehicle Systems, SAE J3061 [5], provides guidance for threat identification and assessment, and guidance for security

in the development process. HEAVENS [6] and EVITA [8] are two Threat Analysis and Risk Assessment (TARA) models also referred to in SAE J3061, which result in security levels and methods for how to specify security requirements for identified threats. Later in this paper, we apply the HEAVENS model to our automotive use case, to identify security relevant items, i.e., functions, components and messages (signals), and classify their security needs. The resulting security levels constitute of six elements representing the security attributes mitigating the STRIDE threats: *integrity*, *authenticity*, *non-repudiation*, *confidentiality*, *availability*, and *authorization*.

IEC 62443 [9], a security standard for industrial communication networks, performs a similar mapping between system requirements and security levels. IEC 62443 is not entirely applicable for the automotive domain due to the focus on the human operation of industrial automation systems.

The NIST FIPS PUB 199 [10], *Standards for Security Categorization of Federal Information and Information Systems*, describes a security classification for information systems using *confidentiality*, *integrity*, and *availability* with each having three levels, low, moderate and high. Additionally, the NIST special publication SP 800-53 [11], *Security and Privacy Controls for Federal Information Systems and Organizations*, contains security mechanisms for information systems. The work in the *Connected Vehicles Pilot Development Phase 1 - Security Management Operating Concept – New York City* [12] follows both, the NIST FIPS PUB 199 and NIST SP 800-53, and identifies safety and privacy requirements for specific usage scenarios and further divides these requirements in information flow and device classes. As many of the NIST SP 800-53 security controls are not appropriate for the automotive domain, the identified security requirements in this project do not cover the requirements for the automotive domain in depth.

The United Nations Economic Commission for Europe (UNECE) has started a task force on cybersecurity and over-the-air issues in [13]. The version from 27/11/2017 consists of a reference architecture, a list of security principles and security controls to mitigate certain threats. We have taken security mechanisms and requirements from all these works into account when selecting relevant mechanisms applicable for the automotive domain.

3 Security Mechanisms and Design Rules

Security mechanisms are used to mitigate threats and to minimize the risk of security attributes of an item being violated. Table B.1 shows the mapping between the STRIDE threats and the corresponding security attributes, including an explanation from HEAVENS [6, p.6]. In Table B.1 we additionally associate the STRIDE threats with item types to emphasize the threats violating the security attributes related to these types. For example, mechanisms that mitigate all types of threats are necessary in order to properly protect messages or signals – the information flow.

The benefits of having a direct relation between STRIDE and security attributes is that any threat assessment model can be used to derive required mandatory security requirements provided a mapping to STRIDE exists. For example, a mapping of the CIA model in [10] to STRIDE is also possible, however, the desired granular-

Table B.1: Mapping of STRIDE threats to security attributes and types of items that need to be protected: information flow, message (msg), firmware (fw), and hardware (hw)

STRIDE Threat [4]	Security Attribute* (from [5], [6])	Explanation [6]	Item Type
Spoofing	Authenticity	Attackers pretend to be someone or something else	MSG, FW, HW
Tampering	Integrity	Attackers change data in transit or in a data store	MSG, FW, HW
Repudiation	Non-repudiation	Attackers perform actions that cannot be traced back to them	MSG, FW, HW
Information disclosure	Confidentiality	Attackers get access to data (e. g., in transit or in a data store)	MSG, FW
Denial of Service	Availability	Attackers interrupt a system's legitimate operation	MSG, FW
Elevation of privilege	Authorization	Attackers perform actions they are not authorized to perform	MSG, FW, HW

* Unlike SAE J3061 [5], we include the attribute freshness in non-repudiation and omit privacy, as we believe that privacy needs to be addressed separately.

ity decreases due to the aggregation of security attributes. A proposed mapping between security levels and identified mechanisms is provided in Table B.2. For some identified design rules and mechanisms, it is favorable to divide them into different classes represented as numbers instead of dots in the table describing the requirements in more detail. For instance, the requirement *AU.3 Verify authenticity of firmware/functions on start* has class 1 *on demand verification of modules* and a stricter class 2 *secure boot*. Furthermore, we highlight that these security mechanisms are mandatory, nevertheless, a mechanism may be disregarded when properly argued.

The following sections describe the six security attributes and list the corresponding security mechanisms and design rules. The security levels the mechanisms are associated with range from 0 to 4, where level 0 demands no security and level 1 and upwards require increased security needs.

3.1 Integrity

Integrity is a property that ensures that data, like messages and firmware, have not been altered due to random errors during transmission or by a malicious node. Message Authentication Codes (MACs) can be used to verify the integrity of data. We propose the use of MACs with securely stored pre-shared keys as the required

minimum to provide message integrity. Verifying the integrity of software during boot or when upgrading also requires the use of cryptographic hashes along with secret keys. In order to decrease the time overhead the verification of the firmware takes, the firmware may be partitioned into modules and the verification of the integrity may only be performed for modules currently needed.

IN.1 Message Authentication Code (MAC) with pre-shared key.

IN.2 Cryptographic hash function with pre-shared key to verify *firmware* integrity when upgrading.

IN.3 Cryptographic hash function with pre-shared key to verify *firmware* or *function* integrity on boot.

IN.4 Physical protection against tampering.

IN.5 Detection of physical tampering.

3.2 Authenticity

Authenticity guarantees the origin of data, thus mitigates the possibility to spoof data, i.e., messages and firmware. MACs combined with authentication of nodes and session keys can be used to provide authenticity and integrity of messages. Authenticity of firmware received via over-the-air updates or through a diagnostics device may be achieved with digital signatures and physically unclonable functions may be used to ensure the authenticity of hardware the Electronic Control Unit (ECU) communicates with.

AU.1 Message Authentication Codes (MACs) with session keys to provide message integrity and authenticity.

AU.2 Verify authenticity of firmware when upgrading using digital signatures.

AU.3 Verify authenticity of firmware/functions on boot using digital signatures.

AU.4 Verify authenticity of hardware, e. g., ECUs and diagnostics devices.

3.3 Non-repudiation

In order to cope with repudiation of messages, it is necessary to include counters or timestamps in messages to be authenticated to guarantee freshness. Moreover, it is needed to store logs as evidence of performed transactions and the use of digital signatures to provide proof about the source of the message.

NR.1 Freshness mechanism, e. g., adding a counter or timestamp to message to be authenticated.

NR.2 Audit logging.

NR.3 Use of digital signatures for messages (signals).

3.4 Confidentiality

Encryption provides confidentiality of data. Some microcontrollers can include embedded Hardware Security Modules (HSMs) to provide hardware accelerated encryption and a secure storage of cryptographic keys. Hardware acceleration can be crucial when transmitting and receiving time-critical authenticated and/or encrypted messages. In case decryption is only required for firmware updates, it might be sufficient to rely on software-based methods. Some ECUs might also not have the resources to decrypt the firmware and to perform the update. In such cases it has to be decided whether it is sufficient that the gateway ECU decrypts and verifies the firmware update.

CO.1 Encryption of *messages* when transmitted over the network.

CO.2 Encryption of *firmware* when transmitted over the network.

3.5 Availability

Denial-of-Service (DoS) attacks can target the network and computational resources of an ECU. The effectiveness of DoS attacks aiming at the network strongly depend on the network architecture. The Controller Area Network (CAN) bus, for instance, is due to its nature highly vulnerable to DoS attacks (see [1]), whereas Flexray (ISO 17458-1:2013) is based on time-triggered control for accessing the physical medium. As example, this time-triggered control approach already limits the extent of the attack to the time slots the compromised node is allowed to send. Other targets of DoS attacks can be memory and I/O resources. Watchdog timers can protect against some DoS attacks targeting the computational resources.

AV.1 Limiting access to network resources - Quality of Service (QoS).

AV.2 Use of watchdog timers.

3.6 Authorization and Access Control

Authorization to perform certain tasks or to send specific messages can be enforced individually or by introducing roles on many levels, e. g., network, host, and function-level. Restricting communication between ECUs can be controlled by separating the ECUs into different domains and connecting these domains via gateways that use whitelists. A more radical approach is complete isolation of a specific network segment, which may be feasible in only a few cases. Host-based access control on the other hand can be used to restrict the types of messages and their frequency an ECU is allowed to send. Examples of logical separation are encryption of network traffic, use of Virtual Local Area Networks (VLANs) and virtualization techniques using hypervisors or containers. Domain isolation on the other hand, requires separated physical networks, total isolation from other functions and processes, dedicated memory, and guaranteed computational resources.

AC.1 Whitelisting of messages (signals) on gateways.

AC.2 Whitelisting of messages (signals) on ECUs.

AC.3 Access control on function level.

AC.4 Detection and logging of intrusions.

AC.5 Logical separation.

AC.6 Domain isolation.

Table B.2: Mapping between security mechanisms and security levels.

		SL 1	SL 2	SL 3	SL 4
Integrity	IN.1 [MSG] Message Authentication Code (MAC) with pre-shared key		•	•	•
	IN.2 [FW] Verify cryptographic hash of firmware when upgrading	•	•	•	•
	IN.3 [FW] Verify cryptographic hash of firmware/functions on boot			•	•
	IN.4 [HW] Physical protection against tampering			•	•
	IN.5 [HW] Detection of physical tampering	•	•	•	•
Authenticity	AU.1 [MSG] Message Authentication Code (MAC) with session key			•	•
	AU.2 [FW] Verify authenticity of firmware when upgrading using digital signatures[a]	1	1	2	2
	AU.3 [FW] Verify authenticity of firmware/functions on boot using digital signatures[a]			1	2
	AU.4 [HW] Verify hardware authenticity				•
Non-repudiation	NR.1 [MSG] Freshness using counter or timestamp in authenticated message			•	•
	NR.2 [MSG] Audit logging			•	•
	NR.3 [MSG] Use of digital signatures for messages (signals)				•
Confidentiality	CO.1 [MSG] Encryption of messages			•	•
	CO.2 [FW] Encryption of firmware during transmission[a]			1	2
Availability	AV.1 [MSG] Limited network access – Quality of Service			•	•
	AV.2 [FW] Watchdog timer		•	•	•
Authorization and Access Control	AC.1 [MSG] Whitelisting of messages (signals) on gateways	•	•	•	•
	AC.2 [MSG] Whitelisting of messages (signals) on nodes			•	•
	AC.3 [MSG] Access control on function level			•	•
	AC.4 [MSG] Deployment of Intrusion Detection Systems			•	•
	AC.5 [MSG, FW, HW] Logical separation[a]		1	1	2
	AC.6 [MSG, FW, HW] Domain isolation			•	•
Other requirements[b]	OR.1 Fail in known state				
	OR.2 Information Input Validation				
	OR.3 Operate with least set of privileges that are necessary				
	OR.4 Compliance to secure coding guidelines				
	OR.5 Secure Logging				

[a] The numbers imply the class of a mechanism – higher numbers imply higher demands.
[b] These requirements have not been mapped to security levels as they are either required by laws and regulations, should be considered when developing secure systems or strongly depend on the application.

3.7 Other Requirements

Some security design rules or guidelines have to be fulfilled regardless of the security level due to compliance with laws and regulations or strongly depend on the application rather than a specific security level. Compliance to safe and secure coding guidelines, such as MISRA C Guidelines [14] or SEI CERT C Coding Standard [15], are the base for safe, secure and reliable software. More design rules are for instance least privilege and fail-safe defaults, which have been already listed by Saltzer and Schroeder back in 1975 in [16]. Logging of errors is essential when deploying Intrusion Detection Systems (IDSs), additionally, all violations risen by security mechanisms should be logged. We strongly recommend to take the requirements below into consideration when designing the system.

OR.1 Fail in known state (safe defaults).

OR.2 Input Validation.

OR.3 Operate with least set of privileges that are necessary.

OR.4 Compliance to secure coding guidelines.

OR.5 Secure logging of errors, data modification and updates.

4 Use Case and Attack Model

The reference architecture for the automotive system we will use in our discussion is shown in Figure B.2 and was developed in the HoliSec project [17]. It illustrates a highly simplified version of ECUs and their functionality where the Vehicle ECU is responsible for the park brake status, warning light and gear requests. The OBD-II port is provided by the Edge Node GW which further interconnects networks to the Infotainment unit, CAN bus ECUs, Ethernet nodes, and the Driver Control unit. The Driver Control unit additionally interconnects sensors, such as radar and cameras, the Vehicle ECU, and the Driver Display.

4.1 Use Case

Figure B.3a illustrates the functions involved when implementing Cruise Control (CC) functionality and what messages (signals) these functions exchange. Function Cruise Control is located in the Vehicle ECU. It receives target speeds from the driver and sends a *VehicleSpeedCommand* to the function Speed Control which is responsible for maintaining the target speed and therefore also listens to *VehicleSpeed* broadcasts from the Vehicle Speed function. In order to maintain the target speed, Speed Control always broadcasts a *BrakeCommand* on the CAN bus. Function Braking listens to *BrakeCommand* and *VehicleSpeed* to determine the best braking strategy to be applied to the engine and, if needed, also engages the foundational brakes. Function Braking broadcasts *BrakeEngagedPercentage* to inform other functions, such as Light Control, about its actions.

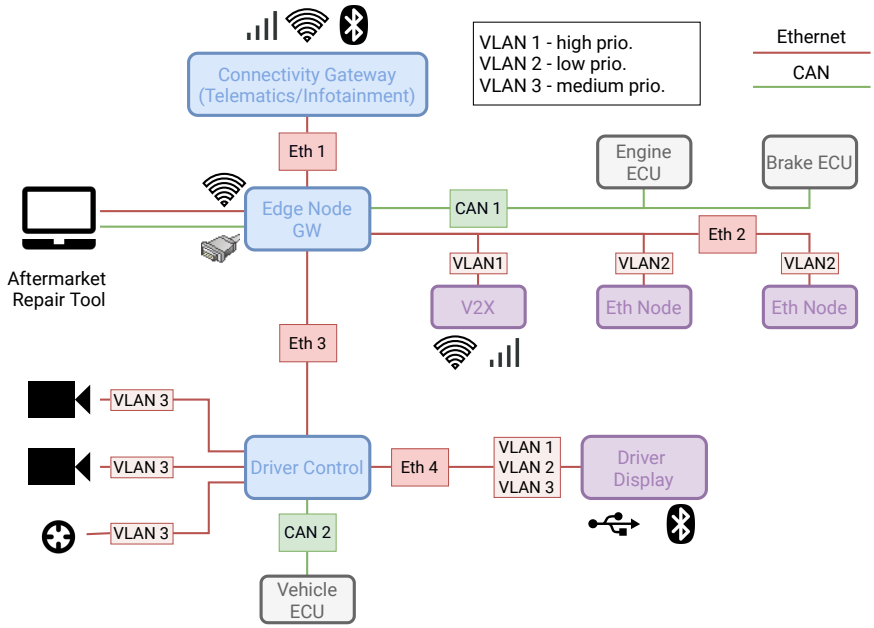
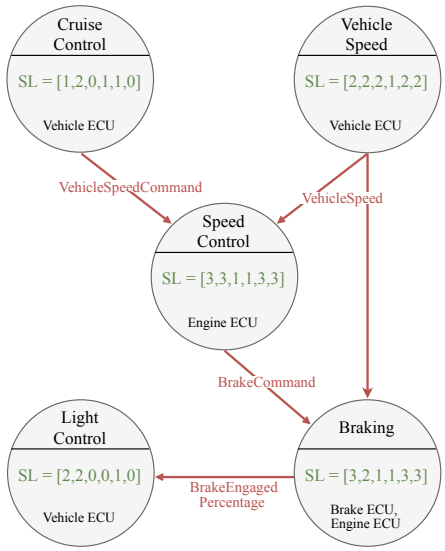


Figure B.2: HoliSec reference architecture of an in-vehicle network.

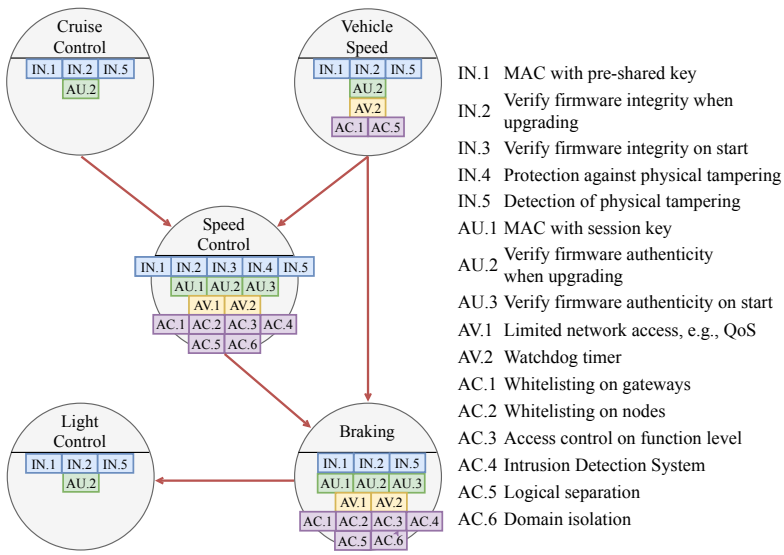
Furthermore, Figure B.3a shows the security levels obtained from a HEAVENS TARA analysis, additionally, we added the security demands on the messages to the source component that is broadcasting information. The results from applying threat assessment techniques may vary for this example, however, the use case is used for illustration purposes to discuss possible security mechanisms in an environment consisting of different functions with different security levels communicating with each other. The security levels shown correspond to the abbreviations of the security attributes from Section 3: $SL = [\text{AU}, \text{IN}, \text{NR}, \text{CO}, \text{AV}, \text{AC}]$.

4.2 Possible Attacks

There are many ways to attack this system, for example injection of faulty or wrong messages and modifications on the functions themselves. We will focus on the two target functions Speed Control and Braking to motivate the results of the TARA analysis and the resulting security levels shown in Figure B.3a. A modification or injection of faulty/wrong *VehicleSpeedCommands* or *VehicleSpeed* messages can cause the Speed Control to think that the vehicle is driving at low speed which results in a high acceleration. Modifications of the Speed Control function in the ECU can lead to dangerous situations if the *BrakeCommand* or the maximum acceleration are being altered. An attack on the Braking function is also dangerous if messages the function listens to are modified which entails a wrong brake behavior and may cause an accident. Moreover, modifications of the function itself can have severe outcomes.



(a) Function view and resulting security levels from TARA analysis.



(b) Required security mechanisms per Function.

Figure B.3: Function view of the Cruise Control (CC) use case.

4.3 Required Mechanisms

Once each function has a security classification, it is possible to find the mandatory security mechanisms according to Table B.2. The required mandatory security mechanisms for this use case are illustrated in Figure B.3b, which shows the required mechanisms as boxes in each function.

It can be seen that functions with security levels up to level 2 only need to fulfill basic security requirements, such as verifying the firmware integrity when upgrading, whereas the functions Speed Control and Braking have higher security demands.

4.4 Applying the Framework

Mechanisms need to be deployed on the ECU and on the network or bus. Mechanism *AC.1 Whitelisting on gateways* needs to be deployed on the Edge Node GW and the Driver Control since they act as gateway between the Vehicle ECU and the Engine ECU. Also note that, if the source function requires *AU.1 MAC with session key*, the receiving nodes need to fulfill AU.1 for these messages. *AV.1 Limited network access* requires a change in the underlying network technology, as CAN itself does not provide means to limit the network traffic per sender. In this case one may argue that alternative security measures on the gateway, such as firewall-like functionality to handle DoS attacks or physical separation, in combination with *IN.4, IN.5, AU.2 and AU.3* are sufficient for this specific case. Another example is mechanism *AC.5 Logical separation*, it may be sufficient to use VLANs for class 1, class 2 on the other hand requires other virtualization techniques that are affecting all other functions realized on the same ECU.

Combining items inside one ECU or even within one subnet requires the aggregation of security levels by choosing the highest occurring security level for each element in the vectors. Nevertheless, it is up to the system designer to choose the level of aggregation that provides the best trade-off between detail and performance/hardware requirements.

5 Discussion

This is a first attempt towards a mapping between security levels and required security mechanisms for the automotive domain. We strongly believe that having a standardized and mandatory way to select required security mechanisms based on the security level of the component is the next step in vehicular security. There exist many models for how to assess threats which result in a security classification of components, and we continue at this point and identify suitable security mechanisms for each security level.

Strict rules are necessary since many parties are involved in the design and development process. Vehicle manufacturers develop some parts of the vehicular system in-house, but many components are provided by suppliers. We believe that such a strict rule-set is necessary in order to not leave the responsibility for developing secure systems to the individual developers or to third-party developers. With this framework in place, all involved parties know the minimum security

measures that need to be implemented and all designers are already provided with basic protection against a large number of threats. Additionally, designers may also add extra application specific requirements that are not covered by the required mechanisms. The decision of not implementing a certain mechanism may arise, due to other restrictions, such as cost and energy consumption. In such a case the choice of using alternative methods needs to be properly justified.

Moreover, this framework enables system designers to easily obtain an overview of the required mechanisms for each item in a bigger system context, making it possible to see dependencies between items, safety-critical or not, at an early stage.

We are aware that our proposed security mechanisms need to be defined in more detail, e. g., requirements on the Hardware Security Module, key derivation methods, encryption algorithms, and key lengths, and that the assigned security levels may need to be adapted, however, we are convinced that a framework similar to what we propose needs to be in place for automotive security. In addition, we show with the Cruise Control (CC) use case how this framework should be applied and we have validated the usefulness and correctness of the identified mechanisms and their corresponding security levels with a large vehicle manufacturer.

6 Conclusion

Threat Analysis and Risk Assessment (TARA) is commonly used to derive security levels which indicate the security demands for components. Current models, such as HEAVENS [6], describe methods to obtain security requirements, however, they do not provide a direct mapping from security levels to security mechanisms and design rules which reduce the feasibility and impact of an attack. In this paper, we have identified appropriate security mechanisms applicable for the automotive domain and consequently associated these mechanisms with security levels that describe the demand for security. Having such a framework in place increases the efficiency and transparency when deriving security requirements for an automotive system, as current approaches place the task of deciding which security mechanisms to implement on the individual designers. We have additionally motivated the proposed mapping with an automotive use case which has been verified with a large vehicle manufacturer.

We have listed the identified mandatory security mechanisms required for specific security levels in Table B.2. Some components may require stricter rules in order to fulfill the demand on security but having a framework like we propose in place covers already basic security requirements that must be implemented.

The interaction between vehicle manufacturers and third-party developers of modules and ECUs benefits from such a framework as well, since all involved parties will have the same expectations of what has to be implemented for each component.

We believe that this framework is a first step towards a standardized mapping to security mechanisms. Such a framework is necessary, as it makes security design easier for system designers, developers, suppliers and all other involved parties. In addition, it prevents individual developers from making poor security design choices since the components have to provide the required security mechanisms.

In future work, this framework needs to be validated with more use cases and security mechanisms have to be specified in more detail. However, a task such as defining the specific algorithms to be used has to be performed by standardization organization in form of regularly updated recommendations.

Acknowledgements. This research was funded by the HoliSec project (2015-06894) funded by VINNOVA, the Swedish Governmental Agency for Innovation Systems.

Bibliography

[1] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.

[2] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[3] "ISO 26262:2011 Road Vehicles – Functional Safety," International Organization for Standardization (ISO), Standard, 2011.

[4] Microsoft Corporation, "The stride threat model," 2005, (Accessed: 2017-11-06). [Online]. Available: https://msdn.microsoft.com/en-us/library/ee823878.aspx

[5] "SAE J3061: SURFACE VEHICLE RECOMMENDED PRACTICE - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," SAE International, Standard, 2016.

[6] M. M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson, "A risk assessment framework for automotive embedded systems," in *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security - CPSS 16*. Association for Computing Machinery (ACM), 2016.

[7] "ISO/IEC 27000:2016 Information technology – Security techniques," International Organization for Standardization (ISO), Standard, 2016.

[8] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weylr, "Security requirements for automotive on-board networks," in *9th International Conference on Intelligent Transport Systems Telecommunications, (ITST)*. Institute of Electrical and Electronics Engineers (IEEE), 2009.

[9] "IEC 62443 – Industrial communication networks - Network and system security," International Electrotechnical Comission, Standard, 2013.

[10] "NIST FIPS PUB 199 – Standards for Security Categorization of Federal Information and Information Systems," National Institute of Standards and Technology, Standard, 2004. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.199

[11] "NIST Special Publication 800-53 – Security and Privacy Controls for Federal Information Systems and Organizations," National Institute of Standards and Technology, Standard, 2013. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-53r4

[12] S. Galgano, M. Talas, W. Whyte, J. Petit, D. Benevelli, R. Rausch, and S. Sim, "Connected Vehicles Pilot Deployment Phase 1 – Security Management Operating Concept – New York City," 2016.

[13] UNECE, "TFCS-09-14 Draft Recommendation on Cyber Security of the Task Force on CyberSecurity and Over-the-air issues of UNECE WP.29 IWG ITS/AD," 2017.

[14] *MISRA C: Guidelines for the Use of the C Language in Critical Systems 2012*. Motor Industry Research Association, 2013.

[15] "SEI CERT C Coding Standard Rules for Developing Safe, Reliable, and Secure Systems," Carnegie Mellon University, book, 2016.

[16] J. Saltzer and M. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[17] A. Yadav and C. Sandberg. (2018) Holisec reference architecture. (Accessed: 2018-04-10). [Online]. Available: http://autosec.se/wp-content/uploads/2018/04/HOLISEC_D4.1.3_v1.0.pdf

Chapter C

REMIND: A Framework for the Resilient Design of Automotive Systems

Adapted version that appeared in SecDev 2020

T. Rosenstatter, K. Strandberg, R. Jolak, R. Scandariato, T. Olovsson

Abstract. In the past years, great effort has been spent on enhancing the security and safety of vehicular systems. Current advances in information and communication technology have increased the complexity of these systems and lead to extended functionalities towards self-driving and more connectivity. Unfortunately, these advances open the door for diverse and newly emerging attacks that hamper the security and, thus, the safety of vehicular systems. In this paper, we contribute to supporting the design of resilient automotive systems. We review and analyse scientific literature on resilience techniques, fault tolerance, and dependability. As a result, we present the REMIND resilience framework providing techniques for attack detection, mitigation, recovery, and resilience endurance. Moreover, we provide guidelines on how the REMIND framework can be used against common security threats and attacks and further discuss the trade-offs when applying these guidelines.

REMIND: A Framework for the Resilient Design of Automotive Systems

1 Introduction

In the past years great effort has been spent in publishing guidelines and standards for security frameworks specific to their domains and in identifying security principles. Examples range from the NIST guideline for cybersecurity in smart grids [1], the cybersecurity guideline for ships [2], cybersecurity guidelines for the automotive domain [3–5] and the upcoming ISO/SAE standard for cybersecurity engineering for road vehicles, namely ISO 21434 [6].

Resilience is the next step towards reliable, dependable and secure vehicular systems. Vehicles need to be able to mitigate faults, errors, attacks and intrusions that would ultimately result in failures in order to withstand safety and security threats from their environment. We define automotive resilience as the *"property of a system with the ability to maintain its intended operation in a dependable and secure way, possibly with degraded functionality, in the presence of faults and attacks."* This definition is inspired by Laprie's definition [7] and the definition of network resilience by Sterbenz et al. [8]; however, the chosen definition highlights that faults or changes, e.g., functional and environmental (see [7]), can also be originated by an attacker whose aim is to disrupt the system.

Resilience can be obtained in many different ways and on different levels, i. e., hardware, software or (sub)-system level. Today's internal architecture of vehicles is quite complex and can be distributed over more than hundred so-called Electronic Control Units (ECUs). However, we are currently in a transition towards a more centralized architecture where functions will be concentrated on much fewer and more powerful ECUs [9]. These central ECUs are connected to sensors, actuators, external communication media and to some extent to smaller legacy subsystems. Such a centralized architecture enables vehicle OEMs not only to perform more resource intensive operations needed for autonomous driving, but also allows to introduce new designs and technologies needed to secure and protect these highly connected and autonomous vehicles. Virtualization is seen as one key technology enabling the isolation of vehicle functions from each other along with the possibility to dynamically assign hardware resources. Introducing resilience to such a centralized automotive system requires the deployment of techniques and principles in all layers and components of the system, ranging from the vehicle itself, the connected IT infrastructure, road infrastructure and the communication to other vehicles.

Motivation. The increasing complexity towards autonomous driving combined with the interconnectedness of vehicles, e. g., vehicle-to-vehicle and vehicle-to-infrastructure communication, and the continuous development of functions require

vehicles to react and adapt to changes and attacks independently. The automotive domain is distinct from other domains as it is a safety and real-time critical system operated by millions of individuals each day. Furthermore, security and safety techniques need to be aligned and extended with resilience techniques in order to strengthen vehicles' capabilities to withstand impending threats.

Contributions. This paper provides a framework to design resilient automotive systems. First, we systematically identify relevant automotive resilience techniques proposed in the literature with the goal to provide a full picture of available tools and techniques. We also organize these techniques into a taxonomy, which comprises the categories of Detection, Mitigation, Recovery, and Endurance (REMIND). These categories represent high-level strategies that can help designers understand the *purpose* of each technique. Further, it can be beneficial to combine techniques from different strategies to achieve multiple layers of security. The selection of the right technique for the task at hand is further supported by associating the resilience techniques to the classes of *automotive assets* they are appropriate for. Additionally, we elaborate on the *trade-offs* (i.e., pros and cons) that are associated with each of the techniques, e.g., with respect to performance and other qualities. In summary, we provide a multi-dimensional decision support framework (built in a bottom-up fashion from the analysis of the literature) that can lead designers to the informed and optimal selection of a suitable set of resilience techniques to be implemented in an automotive system.

2 Methodology

By means of a systematic literature survey, we identify research papers that discuss techniques that are suitable to provide automotive resilience. We consider existing work related to resilience, fault tolerance and dependability. We also analyze the papers describing each technique to understand (i) the assets that can benefit from the technique, (ii) the risks that are mentioned as being mitigated by the techniques, and (iii) any pros/cons associated with the use of such technique.

We identified relevant research papers by searching the Scopus database[1]. A search string was intended to find relevant publications that carried out a review of suitable techniques. Therefore, we formulated the search string to *include* survey or literature review, and relevant topics, such as resilience, survivability, attack recovery, error handling or fault tolerance, as well as the keywords software, system or network. We *excluded* the keywords FPGA, memory, wireless, SDN and hardware to limit the search result to publications focusing on system architecture, software design or physical networks. Furthermore, we considered only publications written in English and published after 2010 in the areas of computer science and engineering. We manually screened the 200 most relevant publications returned by Scopus and found eight additional research publications, which were added to our result set. Ultimately, we retained and analyzed 12 publications which are shown in Table C.1.

[1] https://www.scopus.com/

Table C.1: Publications that provide an overview or collection of relevant techniques.

Discipline	Existing Work	Domain
Resilience	Chang2015 [10]	Cloud Computing
	Hukerikar2017 [11]	High Performance Computing
	NIST 800-160v2 [12]	Systems Engineering
	Ratasich2019 [13]	Cyber-Physical Systems
	Sterbenz2010 [8, 14]	Networks
Security	Segovia2019 [15]	SCADA systems
Dependability	Bakhshi2019 [16]	Fog Computing
Fault Tolerance	Egwutuoha2013 [17]	High Performance Computing
	Kumari2018 [18]	Cloud Computing
	Mukwevho2018 [19]	Cloud Computing
	Slåtten2013 [20]	Software Engineering
	Wanner2012 [21]	Vehicle Controller

3 Attack Model and Assets

The four *strategies* in the REMIND framework are, as shown in Figure C.1, further refined in *patterns* and *techniques*. A collection of these techniques specific for automotive systems is described in Section 4 and has been identified based on existing research in other domains and areas (see Table C.1). We additionally describe the trade-offs of these techniques in Appendix C.A and point to relevant publications in Appendix C.B. In the remainder of this section we describe the assets, security threats and attacks of automotive systems.

We consider four asset types, namely *Hardware*, *Software*, *Network/Communication* and *Data Storage*. The attacker aims to compromise these assets via various attack vectors, whereas the defender, i.e. the vehicle, aims to cope with these attacks via resilience techniques. We consider skilled attackers as well as novice hackers (e. g., script kiddies) and further give examples from an asset, threat and attacker perspective.

Hardware. Can be broken down to *ECUs*, *Sensors* and *Actuators*. An *ECU* can vary in complexity depending on its objective, from a specific limited task to a multitude of tasks. The former can relate to the processing of a sensor signal and the latter an infotainment-system with lots of applications. *Sensors* can give information about speed, temperature and obstacle distance and identification where the *Actuators* turn input from these sensors (via an ECU) into actions, such as braking, steering and engine control.
Attack example. Tampering with existing hardware or installing malicious hardware into the vehicle can act as mediators to gain complete vehicle control. Input signals from sensors may be manipulated to cause an unwanted behavior.

Software. Can be *in transit*, *at rest* or *running*. *In transit* can relate to software provisioning systems, such as over-the-air or workshop updates and the latter two to software installed or running in ECUs.
Attack example. Software vulnerabilities might be exploited, e. g., via a privilege escalation attack which enables ECUs to be re-programmed with additional functionalities, such as adding remote access to the system.

Network/Communication. Can be broken down to *internal* and *external communication*. Examples for internal communication are CAN, FlexRay, LIN, MOST and Automotive Ethernet and for external communication Wi-Fi, Bluetooth, and V2X as well as external interfaces such as OBD-II, debug ports (e. g., JTAG) and CD player.
Attack example. The attacker can try to inject malicious data, through a device connected to an in-vehicle bus affecting the internal communication. Furthermore, modification of V2X data from other vehicles as well as malicious roadside units (e. g., vehicle positioning or traffic condition data) could affect system functions.

Data Storage. Can potentially be sensitive data, such as cryptographic keys, forensics logs, system information (e. g., from software libraries, OS and applications) and reports about the vehicle and the driver.
Attack example. The attacker can exploit secret keys used for sensitive diagnostics to disable firewalls. Logs and report data might be manipulated or removed to hide forensic evidence of the crime. Furthermore, information about the system can reveal vulnerabilities which might be exploited.

Attackers typically exploit the above-mentioned assets in any order to achieve their goal, e. g., uploading malicious software to the vehicle by first compromising the cryptographic keys to get access to the memory and consequently upload a modified firmware containing malicious code. This can give elevated privileges and extended functionality which could cause inconsistencies or disruption of the system.

More examples of assets and related security threats and attacks can be found in Table C.2.

4 REMIND Automotive Resilience Framework

We have developed the REMIND framework shown in Figure C.1 to provide system designers and developers with a categorization of suitable resilience mechanisms including the identification of the assets they protect. The structure of the layers is chosen similarly to the work in Hukerikar et al. [11], where the bottom layer is divided into *strategies* and the mid layer is split into *patterns* that provide more details about the way the strategies can be realized. We refer to relevant solutions for automotive systems in the top layer and further link to the survey papers and reviews that identify specific *techniques* for their domain in the description listed below.

The four REMIND strategies for providing resilience for vehicular systems are:

- **Detection.** Faults, attacks and other anomalies need to be detected by the system in order to take reactive measures to avoid a failure.
- **Mitigation.** Once an anomaly is detected and located, mitigation techniques need to be triggered to keep the system operational. These techniques may result in a non-optimal system state.
- **Recovery.** Transitioning back to the desired, i. e., optimum state, is the aim of recovery.

Table C.2: Automotive assets and related security threats and attacks

Asset	Asset Examples	Security Threat	Attack Examples
Hardware	ECU (hardware) Sensors Actuators	Disruption or direct intervention. Availability and Integrity.	*Fault Injection:* fuzzing, DoS, microprobing, malicious hardware as well as environmental injections (e. g., voltage and temperature) can disrupt or disable components or system resources. *Information Leakage:* side channel parameters, such as timing information or power consumption (e. g., differential power analysis) to extract secret keys.
Software	ECU (software) Libraries OS Virtualization	Manipulation of software, measurements or control signals. Availability and Integrity.	*Malware/Manipulated software:* indirectly affecting storage through alteration, deletion or blocking data, or indirect affecting the communication by read, manipulate or replay of messages, hence causing disruption and deviations from normal system operation.
Network/ Communication	CAN LIN MOST FlexRay Automotive Ethernet Mobile Network Wi-Fi Bluetooth OBD-II CD player	Communication failure or protocol vulnerabilities. Confidentiality, Integrity, Availability and Privacy.	*Fabrication/Jamming attack:* introducing fake traffic, e. g., sending high priority messages, to block legitimate low priority messages. *Masquerading/Spoofing attack:* masquerading as a legitimate node, e. g., by suspending the authentic ECU and send fabricated messages which seems to origin from the same. *Collision:* spoofing a message to induce a bit error/collision and then potentially spoof additional messages which get accepted. *Eavesdropping/hijacking:* intercept to read, block, manipulate or replay messages. *Suspension/DoS attack:* disable an ECU, such as inducing programming mode causing an ECU to not transmit or relay messages, potentially causing other ECUs to malfunction.
Data Storage	User Data Logs/Reports/Events Checkpoints Backups Forensics data Cryptographic material	Malicious handling of data storage. Confidentiality, Integrity, Availability and Privacy.	*Unauthorized read:* acquire sensitive data, such as privacy related user data e. g., previous locations or driving behavior. *Manipulation*: malicious alteration of data, e. g., replacing the software validation key enables potential alteration of memory data. *Removal:* data deletion of sensitive information, such as forensics data. *Reverse engineering:* extraction and analysis of firmware to deduce design features, vulnerabilities or secret keys.

- **Endurance.** The focus is set on lasting resilience in contrast to recovery & mitigation strategies which aim at taking immediate measures.

The remaining part of this section details the strategies and describes the patterns and corresponding techniques.

4.1 Detection

The monitoring and detection capabilities of a system can be limited due to various factors, such as computational resources, energy consumption, and the complexity of functions and network architecture. The move to a more centralized architecture, however, paves the way for more extensive monitoring.

4.1.1 Specification-based Detection

Malicious or abnormal behavior is detected using a specification that describes the behavior of signals or communication patterns. Domain knowledge is needed to create the specifications.

- *Signature-based Detection [13].* Signatures are constructed to describe known attack behavior. By design, these techniques suffer from detecting new attacks and zero-day vulnerabilities. However, they typically achieve a low false positive rate [22].
- *Runtime Verification [13, 20].* A monitor observes the system at runtime to verify the correctness of the execution. Formal specification languages, e. g., Signal Temporal Logic [23], have been developed to describe the normal system behavior which is matched against a trace during execution.
- *Falsification-based Analysis [13].* It extends STL by including a quantitative semantics allowing the return of real values rather than Boolean values.
- *Verification of Safety-Properties [13].* The formal verification of safety properties has become increasingly complex due to the added functionality in modern vehicles. Exhaustive verification techniques, as listed and argued by Ratasich et al. [13], are currently limited to small scale models.
- *Specification-based Anomaly Detection.* Normal behavior, according to a set of rules, is defined using this technique. An alert is sent when a violation of these rules is detected [22].

4.1.2 Anomaly-based Detection

Anomaly- or behavior-based detection techniques are based on comparing behavior with a model of normal behavior. Alerts are raised when a deviation is detected [24].

- *Statistical Techniques [13].* A statistical model describing the system or a specific process is designed in order to detect anomalies. Events are considered anomalies when the probability of their occurrence is below a certain threshold according to the model.

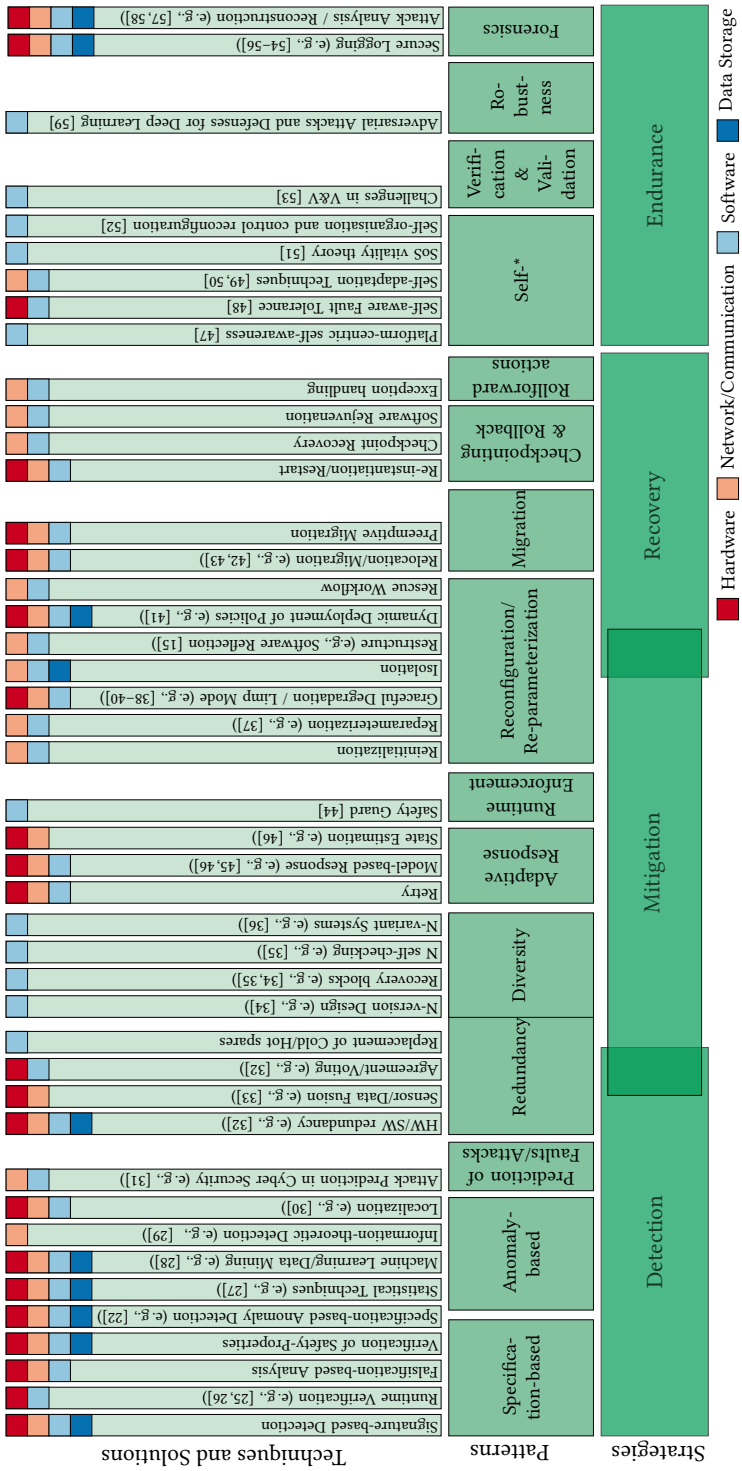


Figure C.1: REMIND resilience techniques and solutions including a mapping to the assets for each technique. The overlap of the *Mitigation* strategy highlights that some patterns also contribute to *Detection* respectively *Recovery*.

- *Machine Learning/Data Mining [13].* These techniques typically do not require domain knowledge. A model, such as Bayesian networks, neural networks and support vector machines, learns through training data how to classify observations in normal and abnormal classes.

- *Information-theoretic Detection [13].* The entropy of information can be used to detect anomalies, as a change of the entropy above a certain threshold may be caused by an attack, e. g., masquerading attack [13, 29].

- *Localization.* Finding the source of the attack may be required to take appropriate actions. Network-based Intrusion Detection Systems (IDSs) can be used to limit the location to a specific subnet, however, solutions identifying the particular ECU are needed (e. g., [30]).

4.1.3 Prediction of Faults and Attacks

First, the system needs to identify the presence of an attacker. The next actions are *attack projection* and *attack intention recognition* which aim at identifying the next steps and the ultimate goal of the attacker. *Attack or intrusion prediction* can be used to foresee when and where an attack will take place [31].

Adversaries mounting simpler attacks on a single vehicle, such as DoS attacks on the CAN bus, may be difficult to predict as the attack consists of fewer steps. However, large-scale attacks requiring the attacker to go through several stages may be predicted by this technique.

4.1.4 Redundancy

Redundancy is twofold, as it can support both detection and mitigation. It is important to highlight that purely redundant systems suffer from the same design faults and vulnerabilities. Thus, diversity is combined with redundancy to overcome this issue.

- *HW/SW Redundancy [11–13, 15, 17–20].* Redundancy combined with a voter allows to mask system failures. The voter compares the results of a number of independently executed software and/or hardware modules and selects, for instance, the majority [32]. Repeating the computation n times on the same hardware can be used to detect random faults.

- *Sensor/Data Fusion [13].* Data from different origins may be fused to compensate inaccuracies or temporary sensor failures. Sensor fusion, e. g., extended Kalman filter [60] and particle filter [33], can be used to describe the non-linear relationship between sensors. For example, the motion of a vehicle can be described with measurements from the wheel speed sensor, GPS location and data received from other vehicles.

- *Agreement/Voting [11, 13, 17, 20].* Redundant components are required for this technique. Voting can be realized in two ways, i. e., exact voting and inexact voting, where the latter allows a variation of the result within a certain range [32].

- *N-version Design [11–13, 17, 19].* N versions of a software with the same requirements are developed by N independent teams resulting in a diverse set of functionally equivalent software components that fulfill the same specification. These versions are executed concurrently and a voter decides based on the majority or calculates, for instance, the median or average of the results [34].

- *Recovery Blocks [11, 19, 20].* Similar to n-version design, n versions of a software component exist; however, only one version is executed at a time. After the active version is executed, a common acceptance test decides whether the result is accepted. In case the result is rejected, the subsequent version is executed and evaluated [34, 35].

- *N self-checking [17].* This technique is a combination of n-version design and recovery blocks. It requires at least two diverse versions with their own acceptance test. When the active component fails its acceptance test, the subsequent component takes over [35].

- *N-variant Systems.* Multi-variant execution automatically diversifies software and monitors the output of at least two variants to detect and mitigate attacks [36].

- *Replacement of Cold/Hot Spares [13].* Concurrent and sequential execution of redundant software components is costly in terms of energy consumption and computational resources. Therefore, the introduction of cold or hot spares, such as in N self-checking, have been found to be a viable alternative [13].

4.2 Mitigation

After detecting an attack or anomaly, the system needs to react to reduce the impact of the attack. Some mitigation techniques may require the transition to a non-optimal state.

4.2.1 Adaptive Response

We focus on techniques that adapt the response of a function or sub-system in order to maintain its intended functionality.

- *Retry [18, 19].* Performing the same computation with new measurements if the first computation resulted in an undesired system state or in an error. Retry can mitigate a replay attack.

- *Model-based Response and State Estimation [15, 21].* System models, e. g., Kalman filter for state estimation [60, 61], or parameter estimation techniques, like regression analysis, are not only a temporary solution to mitigate attacks, such as replay and masquerading attacks, they can also be used to alert the system and log important information for forensics [46].

4.2.2 Runtime Enforcement

Runtime enforcement is an extension of runtime verification where the system also reacts to violations [25].

4.2.3 Reconfiguration and Reparameterization

The system protects itself by adapting parameters when an attack is detected. We distinguish between reconfiguration and migration in the way that migration focuses on relocating functionality whereas reconfiguration changes system or application parameters.

- *Reinitialization [11].* Temporary faults and attacks can be addressed with this technique. However, permanent faults or reoccurring attacks cannot be mitigated by restoring the system or a function to its initial state. Reinitialization can be seen as checkpoint recovery with the checkpoint being the initial state of the system or function.

- *Reparameterization [13].* Is similar to reinitialization, however, the system configuration is dynamically adjusted to the situation. As Ratasich et al. [13] point out, reparameterization typically results in a non-optimal state.

- *Graceful Degradation / Limp Mode [13, 15].* Given the extended automated driving functions of future vehicles, it is of utmost importance to implement more sophisticated solutions that ensure the passengers safety when key components in the vehicle fail or are subject to attacks. These techniques are similar to reparameterization, but focus on safety and should be seen as a last resort. Modern vehicles already have a so-called limp mode implemented, which is triggered when the vehicle detects major technical problems [62].

- *Isolation [11, 13].* Restricting access or completely isolating system components in the presence of an error or intrusion can limit the impact on the entire system and its performance.

- *Restructure [11].* Restructuring components within a sub-system aims at providing resilience through reconfiguration of affected components. Segovia et al. [15] explore software reflection as means to mitigate attacks.

- *Dynamic Deployment of Policies [15].* Security or other policies can be applied dynamically based on the type of attack, e. g., DoS or masquerading, that is detected.

- *Rescue Workflow [18, 19].* A workflow can be used to describe tasks with their dependencies to each other. The idea behind rescue workflows is to dynamically adjust the structure of the workflow when an error or intrusion affects a specific task. Existing cloud solutions may need to be adapted for automotive systems.

4.3 Recovery

Recovery techniques intend to bring the system back to an optimal state.

4.3.1 Migration

These techniques are mainly originating from high performance computing and cloud systems. As future automotive systems move towards a centralized architecture, virtualization and service-oriented architectures are becoming more relevant.

- *Relocation/Migration [13, 19].* Virtualization such as hypervisor and container-based solutions allow a fast migration and relocalization to other nodes in the vehicular network.
- *Preemptive Migration [18, 19].* Continuous monitoring and analysis of the system can be used to relocate software functions or services before a fault occurs.

4.3.2 Checkpointing & Rollback

A checkpoint or snapshot describes the system state at a specific point in time. By design, recovery does not prevent the same attacks from happening again.

- *Re-instantiation/Restart [11, 13, 17, 19].* When an intrusion is detected, the affected component can be re-instantiated or restarted to recover to a known, error and attack free, state. This technique can be combined with reparameterization to avoid the same anomaly to happen again [13].
- *Checkpoint Recovery [11, 17–20].* Snapshots can be created in two ways: checkpoint-based and log-based. Egwutuoha et al. [17] highlight the complexity of taking checkpoints in a distributed system, as these checkpoints need to be consistent.
- *Software Rejuvenation [11, 19].* This technique carries out periodic restarts or reinitializations of the system to maintain a known, error-free state.

4.3.3 Rollforward actions

These techniques aim at bringing the system to a stable state immediately before the error or attack was detected. As in rollback, the recovery is based on using checkpoint-based or log-based recovery [11].

- *Exception Handling [11].* From a model-driven engineering view, *Rollforward* can be performed using exception handling. Slåtten et al. [20] highlight that this solution can be only applied to anticipated events.

4.4 Endurance

Resilience needs to be ensured over the entire lifetime of a vehicle. The preceding techniques center around providing immediate response when anomalies are detected.

4.4.1 Self-*

Self-* or self-X techniques cover solutions and research directions focusing on how to introduce autonomy into the system. This pattern is especially important for future vehicles as the environment is and will change frequently, new vulnerabilities will be found, new attempts to attack vehicles and their infrastructure will be developed, and new technologies will appear. Also, considering the lifetime of cars, which is around 10–15 years, it is evident that automotive systems need to adapt to a certain extent autonomously.

4.4.2 Verification and Validation

Due to the increasing functionality and interconnectedness of modern vehicles it is required to update software components via over-the-air updates in order to fix vulnerabilities and bugs or upgrade vehicle functions. This is especially challenging as each vehicle model can be further configured, resulting in a manifold of possible vehicle configurations.

4.4.3 Robustness

Artificial intelligence, especially machine learning, is a key technology for autonomous driving and decision making, as the system needs to be able to handle previously unseen situations [13].

4.4.4 Forensics

Providing evidence of intrusions even after a crash is important for taking appropriate countermeasures.

- *Secure Logging.* Hoppe et al. [54] express the need for forensic solutions in vehicles. Non-safety-critical events, such as updates, component failures and other malfunctions, need to be logged and stored securely for a prospective analysis. The authors also discuss in great detail which information and how this information can be stored in vehicles.
- *Attack Analysis.* Nilsson and Larson [57] specify requirements for forensic analyses of the in-vehicle network. It is also important to analyze attacks disclosed by researchers, such as Checkoway et al. [63] and Miller and Valasek [64], as well as attacks logged by the vehicle manufacturers in order to take appropriate actions.

5 Related Work

Making vehicles safe and secure has traditionally been the main focus in research. For instance, methods to combine safety and security [65] and how to assess an automotive system and/or derive security requirements and mechanisms have been

proposed [66–68]. Le et al. [69] provide a survey on security and privacy in automotive systems and further provide an overview of suitable security mechanisms.

One of the first structured collections of principles for cyber resilience is the Cyber Resiliency Engineering Framework [70] by MITRE in 2011 which got further incorporated in NIST SP 800-160v2 [12]. Other work describing principles for resilience have been either concentrating on other domains, i. e., high performance computing, cyber-physical systems, or networks, or they focused particularly on dependability or fault tolerance. Table C.1 provides an overview of relevant publications, which provide a comprehensive overview or collection of techniques, and categorizes them according to their discipline and the area they are focusing on.

The reviewed publications classify the identified techniques in different ways. Hukerikar et al. [11] divide them into strategies, i. e., fault treatment, recovery, and compensation, whereas Ratasich et al. [13] organize them according to their ability, i. e., detection and diagnosis, recovery or mitigation, and long-term dependability and security. Work focusing on fault tolerance either split the identified techniques in reactive and proactive measures [18, 19] or classify them according to their ability, e. g., error handling and recovery [17, 20].

With the developed REMIND framework, we contribute to supporting the resilience of automotive systems by: (i) identifying techniques for attack detection, mitigation, recovery, and resilience endurance; (ii) organizing the techniques into a taxonomy to guide designers when selecting resilience techniques; (iii) providing guidelines on how the REMIND framework can be used against common security threats and attacks; and (iv) discussing the trade-offs when applying the techniques that are highlighted in this framework.

In addition to the identified techniques in Figure C.1, we point to implementations relevant for or specific to the automotive domain in Appendix C.B.

6 Conclusion

The reviewed work shows the current research efforts towards making systems resilient to attacks and faults in related domains. We present a novel structure for categorizing resilience techniques in the form of the REMIND framework with the aim to lead designers in making informed decisions when choosing resilience techniques. We build upon the existing work and set the focus on the limitations of automotive systems and their challenges. The REMIND techniques have been chosen considering automotive assets and related attacks which are described in Section 3 and further linked to the guidelines and trade-off analysis in Appendix C.A.

Future work includes the validation of the REMIND framework in regard to studying its applicability in industry in more depth. Furthermore, specific solutions for the identified techniques that consider the unique properties of automotive vehicles can be explored. Especially, the role of software-defined networking and its contribution to resilience can be investigated.

Acknowledgments. This research was supported by the CyReV project (2018-05013) funded by VINNOVA, the Swedish Governmental Agency for Innovation Systems.

Bibliography

[1] "NISTIR 7628 Rev 1 – Guidelines for smart grid cybersecurity," National Institute of Standards and Technology, Tech. Rep., Sep. 2014. [Online]. Available: https://doi.org/10.6028/NIST.IR.7628r1

[2] "The Guidelines on Cyber Security Onboard Ships," BIMCO, CLIA, ICS, INTERCARGO, INTERMANAGER, INTERTANKO, IUMI, OCIMF and WORLD SHIPPING COUNCIL, Tech. Rep., Dec. 2017. [Online]. Available: https://www.ics-shipping.org/docs/default-source/resources/safety-security-and-operations/guidelines-on-cyber-security-onboard-ships.pdf

[3] "Cyber Security and Resilience of smart cars," The European Union Agency for Network and Information Security (ENISA), Tech. Rep., 2016.

[4] "SAE J3061: SURFACE VEHICLE RECOMMENDED PRACTICE - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," SAE International, Standard, 2016.

[5] UNECE, "TFCS-09-14 Draft Recommendation on Cyber Security of the Task Force on CyberSecurity and Over-the-air issues of UNECE WP.29 IWG ITS/AD," 2017.

[6] "ISO/SAE 21434 Road Vehicles – Cybersecurity Engineering," International Organization for Standardization (ISO), Standard, 2020.

[7] J.-C. Laprie, "From dependability to resilience," in *38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks*, 2008, pp. G8–G9.

[8] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245 – 1265, 2010, resilient and Survivable networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610000824

[9] Ö. Andersson, "The Car - A Computer on Wheels," URL: https://www.icse2018.org/getImage/orig/The+Car+%E2%80%93+computer+on+wheels.pdf, May 2018, visited on 2020-05-21.

[10] V. Chang, M. Ramachandran, Y. Yao, Y.-H. Kuo, and C.-S. Li, "A resiliency framework for an enterprise cloud," *International Journal of Information Management*, vol. 36, no. 1, pp. 155 – 166, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S026840121500095X

[11] S. Hukerikar and C. Engelmann, "Resilience design patterns: A structured approach to resilience at extreme scale," *arXiv preprint arXiv:1708.07422*, 2017.

[12] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid, "Developing cyber resilient systems:: a systems security engineering approach," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 800-160v2, Nov. 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v2.pdf

[13] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci, "A Roadmap Toward the Resilient Internet of Things for Cyber-Physical Systems," *IEEE Access*, vol. 7, pp. 13 260–13 283, 2019.

[14] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Redundancy, diversity, and connectivity to achieve multilevel network resilience, survivability, and disruption tolerance invited paper," *Telecommunication Systems*, vol. 56, no. 1, pp. 17–31, 2014.

[15] M. Segovia, A. R. Cavalli, N. Cuppens, and J. Garcia-Alfaro, "A study on mitigation techniques for scada-driven cyber-physical systems (position paper)," in *Foundations and Practice of Security*, N. Zincir-Heywood, G. Bonfante, M. Debbabi, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2019, pp. 257–264.

[16] Z. Bakhshi, G. Rodriguez-Navas, and H. Hansson, "Dependable Fog Computing: A Systematic Literature Review," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2019, pp. 395–403.

[17] I. P. Egwutuoha, D. Levy, B. Selic, and S. Chen, "A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems," *The Journal of Supercomputing*, vol. 65, no. 3, pp. 1302–1326, 2013.

[18] P. Kumari and P. Kaur, "A survey of fault tolerance in cloud computing," *Journal of King Saud University - Computer and Information Sciences*, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319157818306438

[19] M. A. Mukwevho and T. Celik, "Toward a smart cloud: A review of fault-tolerance methods in cloud systems," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[20] V. Slåtten, P. Herrmann, and F. A. Kraemer, "Chapter 4 - model-driven engineering of reliable fault-tolerant systems—a state-of-the-art survey," in *Advances in Computers*, ser. Advances in Computers, A. Memon,

Ed. Elsevier, 2013, vol. 91, pp. 119 – 205. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124080898000045

[21] D. Wanner, A. Trigell, L. Drugge, and J. Jerrelind, "Survey on fault-tolerant vehicle design," *World Electric Vehicle Journal*, vol. 5, no. 2, p. 598–609, Jun 2012. [Online]. Available: http://dx.doi.org/10.3390/wevj5020598

[22] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security*, 2010, pp. 92–98.

[23] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[24] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805 – 822, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128698000176

[25] E. Bartocci and Y. Falcone, *Lectures on Runtime Verification: Introductory and Advanced Topics*. Springer, Cham, 2018, vol. 10457.

[26] D. Heffernan, C. Macnamee, and P. Fogarty, "Runtime verification monitoring for automotive embedded systems using the ISO 26262 functional safety standard as a guide for the definition of the monitored properties," *IET Software*, vol. 8, no. 5, pp. 193–203, 2014.

[27] N. Nowdehi, W. Aoudi, M. Almgren, and T. Olovsson, "CASAD: CAN-Aware Stealthy-Attack Detection for In-Vehicle Networks," *arXiv preprint arXiv:1909.08407*, 2019.

[28] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Canet: An unsupervised intrusion detection system for high dimensional can bus data," *IEEE Access*, vol. 8, pp. 58 194–58 205, 2020.

[29] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 1110–1115.

[30] K.-T. Cho and K. G. Shin, "Viden: Attacker identification on in-vehicle networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1109–1123. [Online]. Available: https://doi.org/10.1145/3133956.3134001

[31] M. Husák, J. Komárková, E. Bou-Harb, and P. Čeleda, "Survey of attack projection, prediction, and forecasting in cyber security," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 640–660, 2019.

[32] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media, 2011.

[33] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[34] L. Chen and A. Avizienis, "N-version programming: A fault-tolerance approach to reliability of software operation," in *Proc. 8th IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-8)*, vol. 1, 1978, pp. 3–9.

[35] J.-C. Laprie, J. Arlat, C. Beounes, and K. Kanoun, "Definition and analysis of hardware-and software-fault-tolerant architectures," *Computer*, vol. 23, no. 7, pp. 39–51, 1990.

[36] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser, "N-Variant Systems: A Secretless Framework for Security through Diversity," in *15th USENIX Security Symposium*, 2006, pp. 105–120.

[37] A. Höller, T. Rauter, J. Iber, and C. Kreiner, "Towards dynamic software diversity for resilient redundant embedded systems," in *Software Engineering for Resilient Systems*, A. Fantechi and P. Pelliccione, Eds. Cham: Springer International Publishing, 2015, pp. 16–30.

[38] T. Dagan, Y. Montvelisky, M. Marchetti, D. Stabili, M. Colajanni, and A. Wool, "Vehicle safe-mode, concept to practice limp-mode in the service of cybersecurity," *SAE Int. J. Transp. Cyber. & Privacy 2 (2)*, feb 2020.

[39] T. Ishigooka, S. Otsuka, K. Serizawa, R. Tsuchiya, and F. Narisawa, "Graceful degradation design process for autonomous driving system," in *Computer Safety, Reliability, and Security*, A. Romanovsky, E. Troubitsyna, and F. Bitsch, Eds. Cham: Springer International Publishing, 2019, pp. 19–34.

[40] A. Reschka, G. Bagschik, S. Ulbrich, M. Nolte, and M. Maurer, "Ability and skill graphs for system modeling, online monitoring, and decision support for vehicle guidance systems," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 933–939.

[41] J. Rubio-Hernan, R. Sahay, L. De Cicco, and J. Garcia-Alfaro, "Cyber-physical architecture assisted by programmable networking," *Internet Technology Letters*, vol. 1, no. 4, p. e44, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.44

[42] Z. Jiang, N. C. Audsley, and P. Dong, "BlueVisor: A Scalable Real-Time Hardware Hypervisor for Many-Core Embedded Systems," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018, pp. 75–84.

[43] P. Alho and J. Mattila, "Service-oriented approach to fault tolerance in cpss," *Journal of Systems and Software*, vol. 105, pp. 1 – 17, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215000643

[44] M. Wu, H. Zeng, C. Wang, and H. Yu, "INVITED: Safety guard: Runtime enforcement for safety-critical cyber-physical systems," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[45] L. F. Cómbita, J. Giraldo, A. A. Cárdenas, and N. Quijano, "Response and reconfiguration of cyber-physical control systems: A survey," in *2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC)*, 2015, pp. 1–6.

[46] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual Reviews in Control*, vol. 32, no. 2, pp. 229 – 252, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578808000345

[47] M. Möstl, J. Schlatow, R. Ernst, N. Dutt, A. Nassar, A. Rahmani, F. J. Kurdahi, T. Wild, A. Sadighi, and A. Herkersdorf, "Platform-Centric Self-Awareness as a Key Enabler for Controlling Changes in CPS," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1543–1567, 2018.

[48] T. D. Nya, S. C. Stilkerich, and C. Siemers, "Self-aware and self-expressive driven fault tolerance for embedded systems," in *2014 IEEE Symposium on Intelligent Embedded Systems (IES)*, Dec 2014, pp. 27–33.

[49] S. Zeadally, T. Sanislav, and G. D. Mois, "Self-Adaptation Techniques in Cyber-Physical Systems (CPSs)," *IEEE Access*, vol. 7, pp. 171 126–171 139, 2019.

[50] D. Weyns, *Software Engineering of Self-adaptive Systems*. Cham: Springer International Publishing, 2019, pp. 399–443. [Online]. Available: https://doi.org/10.1007/978-3-030-00262-6_11

[51] H. Zhang, B. Huang, P. Zhang, and H. Ju, "A New SoS Engineering Philosophy - Vitality Theory," in *2019 14th Annual Conference System of Systems Engineering (SoSE)*, May 2019, pp. 19–24.

[52] G. Vachtsevanos, B. Lee, S. Oh, and M. Balchanos, "Resilient design and operation of cyber physical systems with emphasis on unmanned autonomous systems," *Journal of Intelligent & Robotic Systems*, vol. 91, no. 1, pp. 59–83, 2018.

[53] R. de Lemos, H. Giese, H. A. Müller, and M. Shaw et al., *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–32. [Online]. Available: https://doi.org/10.1007/978-3-642-35813-5_1

[54] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures," *Reliability Engineering & System Safety*, vol. 96, no. 1, pp. 11 – 25, 2011, special Issue on Safecomp 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0951832010001602

[55] S. Lee, W. Choi, J. H. J., and D. H. Lee, "T-Box: A Forensics-Enabled Trusted Automotive Data Recording Method," *IEEE Access*, vol. 7, pp. 49 738–49 755, 2019.

[56] H. Mansor, K. Markantonakis, R. N. Akram, K. Mayes, and I. Gurulian, "Log your car: The non-invasive vehicle forensics," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 974–982.

[57] D. K. Nilsson and U. E. Larson, "Conducting forensic investigations of cyber attacks on automobile in-vehicle networks," in *Proceedings of the 1st International Conference on Forensic Applications and Techniques in Telecommunications, Information, and Multimedia and Workshop*, ser. e-Forensics '08. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[58] W. Bortles, S. McDonough, C. Smith, and M. Stogsdill, "An introduction to the forensic acquisition of passenger vehicle infotainment and telematics systems data," SAE Technical Paper, Tech. Rep., 2017.

[59] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial Examples: Attacks and Defenses for Deep Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[60] G. Welch and G. Bishop, "An Introduction to the Kalman filter," 1995.

[61] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[62] Burdi Motorworks, "Mercedes Limp Home Mode," https://burdimotors.com/2017/11/30/mercedes-limp-home-mode, Accessed 2020-03-16, 2018. [Online]. Available: https://burdimotors.com/2017/11/30/mercedes-limp-home-mode

[63] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.

[64] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[65] G. Macher, E. Armengaud, E. Brenner, and C. Kreiner, "Threat and risk assessment methodologies in the automotive domain," *Procedia Computer Science*, vol. 83, pp. 1288–1294, 2016.

[66] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weylr, "Security requirements for automotive on-board networks," in *9th International Conference on Intelligent Transport Systems Telecommunications, (ITST)*. Institute of Electrical and Electronics Engineers (IEEE), 2009.

[67] M. M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson, "A risk assessment framework for automotive embedded systems," in *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security - CPSS 16*. Association for Computing Machinery (ACM), 2016.

[68] T. Rosenstatter and T. Olovsson, "Towards a Standardized Mapping from Automotive Security Levels to Security Mechanisms," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2018, pp. 1501–1507.

[69] V. H. Le, J. den Hartog, and N. Zannone, "Security and privacy for innovative automotive applications: A survey," *Computer Communications*, vol. 132, pp. 17 – 41, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S014036641731174X

[70] D. Bodeau and R. Graubart, "Cyber Resiliency Engineering Framework (MITRE Technical Report MTR1-10237)," *Bedford, MA: MITRE Corporation*, 2011.

[71] B. Baudry and M. Monperrus, "The multiple facets of software diversity: Recent developments in year 2000 and beyond," *ACM Comput. Surv.*, vol. 48, no. 1, Sep. 2015. [Online]. Available: https://doi.org/10.1145/2807593

[72] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, *Microservices: Yesterday, Today, and Tomorrow*. Cham: Springer International Publishing, 2017, pp. 195–216. [Online]. Available: https://doi.org/10.1007/978-3-319-67425-4_12

[73] C. Engelmann, G. R. Vallee, T. Naughton, and S. L. Scott, "Proactive Fault Tolerance Using Preemptive Migration," in *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2009, pp. 252–257.

[74] R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Design of Software Rejuvenation for CPS Security Using Invariant Sets," in *2019 American Control Conference (ACC)*, 2019, pp. 3740–3745.

C.A REMIND Resilience Guidelines

In this section, we report in Table C.3 resilience techniques that can be used against common security threats and attacks. We also describe the trade-offs when implementing these techniques.

Table C.3: REMIND Resilience Guidelines

Asset Hardware	**Attack** Fault Injection	
Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Detection • Statistical Techniques [13] • Machine Learning/Data Mining [13] • Localization (e. g., [30]) • Sensor/Data Fusion [13]	• Less computation is required. • No domain knowledge is needed. It handles multivariate and non-linear data. • Identifies the exclusive part causing the fault or attack. • Calculates a value of trust of the data sources derived from the normalization factor.	• Very sensitive to outliers, imprecise detection, and increased complexity when modelling non-linear data. • Requires training. Imprecise prediction: false positives and false negatives. Time penalty and resource consumption (power, processing, and storage). • Often applied offline. The precision of the localization is dependent on both, the number of observed parameters and the set frequency for probing monitored resources. • Imprecise detection: false positives and negatives. It also introduces time penalty (increase in execution time) and space penalty (increase in resource usage).
Mitigation • Hardware Redundancy [11–13, 15, 17–20]	• Enables offsetting the effects of faults and attacks, and allows the progress of the system without loss of functionality.	• Time penalty (increase in execution time) and resource consumption (increase in required resources). Hardware costs independent of whether attacks occur. Also, the design and verification of replicas requires an effort.
Recovery • Relocation/Migration [13, 19]	• Maintain system functionality in an operational state as it was before the fault or attack.	• May cause a degraded system, with less functionality, resources, and performance.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Endurance • Self-aware Fault Tolerance [48]	• Enables systems to adapt their behavior when a fault or attack occurs in their environment, thus allowing a continuous operation of these systems.	• Complexity and resource consumption.

Asset	**Attack**
Software	Malware/Manipulated Software

Detection • Signature-based Detection [13] • Runtime Verification [13, 20]	• A precisely calibrated signature effectively identifies abnormal events during software execution. • Well-established and efficient technique to verify the correctness of software execution and monitor the behavior of the system.	• Does not work when designers and 3rd party suppliers (e.g., intellectual property providers) are not trusted. It cannot handle zero-day attacks and, thus, often used in combination with anomaly-based techniques leading to an increased resource consumption and time penalty. • Limited coverage. The used monitoring algorithms usually handle a single execution trace which limits the scope of the verification.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Mitigation • Software Redundancy [11–13, 15, 17–20] • N-Version Design [11–13, 17, 19] • Agreement/Voting [11, 13, 17, 20] • Recovery Blocks [11, 19, 20] • N self-checking [17]	• Helps to contain and exclude malicious behavior (i.e., reduces likelihood of harm). Enable restoration in case of disruption. Enhances the availability of critical capabilities. • Helps to mitigate the impact of failures when a risk is introduced to system design or configuration. • Typically combined with redundancy. Can be used to select, for instance, the average or median of the results provided by the redundant sources. • Uses different implementations of the same design specification to provide tolerance of design faults. • Provides mitigation by creating N versions of the same software, each with its own acceptance test. The version that passes its own acceptance test is selected through an acceptance voting system.	• Resource consumption. It demands the protection of redundant resources. It can degrade over time as configurations are updated or connectivity changes. It is often applied with diversity techniques which increases complexity and leads to scalability issues. • Requires much effort for designing, implementation, testing, and validation of the N independent versions. • Attackers may exploit the voting process in order to force the system to a degraded mode. • Requires extra verification and validation effort and, thus, more resource consumption. It might be difficult to create alternative software implementations without any correlation between the various versions. • Causes an increase in required resources and execution time.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Recovery • Preemptive Migration [18, 19] • Checkpoint Recovery [11, 17–20] • Software Rejuvenation [11, 19]	• Prevents failures from impacting running parallel applications by enabling the migration of running software from one virtual machine to another in real-time. • Helps the system to resume its operation in a state free of the effects of the fault or attack. Frequent checkpointing reduces the amount of lost work. • Helps avoiding the costs of failures from software degradation, as periodic (graceful) restarts of the software component allow the release and re-allocation of memory, thus, operation in a clean state.	• Lack of standardized metrics for measuring and evaluating the health and interfaces between system components. • Overhead in relation to the size and frequency of created checkpoints. Creating a checkpoint, for instance, requires interrupting the normal operation of a system to record the checkpoint. Moreover, it requires storage resources to store the checkpoint. The created checkpoints might potentially contain an error or intrusion that has not been detected yet. Globally consistent checkpoints are not trivial to obtain in a distributed system, due to e. g., variation of the local clock, parallel computation and possible different system states. • Requires shutting the software down and restarting it periodically which causes the software to be unavailable for the duration of the restart. It is often a slow process requiring an extra overhead.
Endurance • Platform-centric Self-awareness [47] • Secure Logging (e. g., [54–56])	• Enables systems to recognize their own state and to continuously adapt to change, evolution, system interference, environment dynamics, and uncertainty. It optimizes resilience, quality of service, and supports system dynamics and openness. It also helps to reduce uncertainties and identify inconsistencies. • Prevents modifying the logs by using e.g., chained hashes. It enables storing security-related events containing information about e.g., flash operations, external interactions, and power downtime. This information helps to reconstruct events, detect intrusions and identify problems.	• Automatically maintaining coherent specifications that capture and monitor security is a challenging task. Complexity, scalability, and difficulty in dealing with uncertainties and inaccuracies. The determination of relevant dependencies in a complex system is also challenging. • Resource consumption and time penalty. Moreover, missing authentication and lack of cryptographic means to ensure data integrity can limit the potential of the logging.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Asset Network/Communication	**Attack** Fabrication/Jamming	
Detection • Specification-based Anomaly Detection (e. g., [22]) ◦ Localization (e. g., [30]) • Verification of Safety-Properties [13]	• Helps detecting anomalies in the system's behavior by reporting the specific deviation that has been observed. ◦ Identifies the exclusive part causing the fault or attack. • Ensures that the system does not evolve in unsafe state starting from some initial conditions.	• Needs of resources for detection and processing of collected information (e.g., costly intelligent sensors). Domain knowledge is required to specify normal behavior. Specifications need to be adapted for each specific vehicle configuration otherwise risk of high false positives or negatives. ◦ Requires additional resources. • It is limited to small scale systems.
Mitigation • Isolation [11, 13] ◦ Restructure [11]	• It provides a remedy to enable the system to continue its operation by offsetting the effect of the attack. Also, it prevents loss of functionality. ◦ Helps to mitigate incorrectness in the interactions between the components or subsystems by excluding the affected part from interacting with the rest of the system, and maintaining system functionality.	• Introduces a time penalty and an increase in required resources (e.g., replica modules that are used to compensate for isolating the affected component of the system). ◦ May cause an operation of the system in a degraded condition which influences its performance and incurs additional time overhead to the system.
Recovery • Relocation/Migration [13, 19] ◦ Re-instantiation/Restart [11, 13, 17, 19]	• Maintain system functionality in an operational state as it was before the fault or attack. ◦ Helps to restore the system to its initial state when the impact of the attack can not be handled in another manner. It guarantees that the impact of the attack is completely removed.	• May cause a degradation in the operation of the system which influences the performance and functionality thereof. ◦ Restoring the system to its initial state causes lost data, such as privacy related data (e. g., location, speed, driving behavior) and workshop data (e. g., vehicle health, engine data and emissions). The impact of the lost data depends on the type of data and the current need for it. In addition, the re-instantiation of safety-critical functions may require the vehicle to be in standstill.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Endurance • Self-adaptation [49, 50]	• Ensures a secure, reliable, and predictable communication between system components and between the system and its environment. Supports and maintains an acceptable level of service despite the occurrence of faults and other factors that affect normal operations. Seamlessly adapts to different network loads and reacts to security threats and other disturbances in the environment.	• Complexity and resource consumption.

Asset	**Attack**
Network/Communication	Masquerading/Spoofing/Collision

Detection • Information-theoretic Detection [13] • Falsification-based Analysis [13]	• Helps to detect anomalies by analyzing available audit logs and records (e.g., entropy measures) and comparing these records with defined normal behaviors. More records enhance the precision of the detection. • Provides an indication (i.e., a robustness degree) to what extent temporal logic properties are from satisfying or violating a specification.	• Time penalty for processing audit records. More records at disposal increases the processing time and complexity. On the other hand, a low number of records leads to an imprecise detection with more false positives and false negatives. • Imprecise detection: false positives and false negatives
Mitigation • Rescue Workflow [18, 19] (adaptation may be necessary) • Dynamic Deployment of Policies [15]	• Enables the system to continue operation after the failure of the task until it is unable to proceed without amending the fault or attack. Already finished tasks do not need re-execution, thus saving time and resources. • Takes the dynamic and changing nature of attacks into account. Deploys different defense policies depending on the attack, for example, it can modify the executed actions while the attack is going on.	• It may lead to a decrease in the quality of service. Time penalty might be caused by recomputing and migrating the tasks which cause the problem. • Leads to performance overhead. Moreover, it always requires runtime permissions which may not be present when running normally. Complexity.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Recovery • Checkpoint Recovery [11, 17–20] • Re-instantiation/Restart [11, 13, 17, 19]	• Helps the system to resume its operation in a state free of the effects of the fault or attack. Frequent checkpointing reduces the amount of lost work. • Helps to restore the system to its initial state when the impact of the attack can not be handled in another manner. It guarantees that the impact of the attack is completely removed.	• Overhead in relation to the size and frequency of created checkpoints. Creating a checkpoint, for instance, requires interrupting the normal operation of a system to record the checkpoint. Moreover, it requires storage resources to store the checkpoint. The created checkpoints might potentially contain an error or intrusion that has not been detected yet. Globally consistent checkpoints are not trivial to obtain in a distributed system, due to e. g., variation of the local clock, parallel computation and possible different system states. • Restoring the system to its initial state causes lost data, such as privacy related data (e. g., location, speed, driving behavior) and workshop data (e. g., vehicle health, engine data and emissions). The impact of the lost data depends on the type of data and the current need for it. In addition, the re-instantiation of safety-critical functions may require the vehicle to be in standstill.
Endurance • Secure Logging (e. g., [54–56])	• Prevents modifying the logs by using e.g., chained hashes. It enables storing security-related events containing information about e.g., flash operations, external interactions, and power downtime. This information helps to reconstruct events, detect intrusions and identify problems.	• Resource consumption and time penalty. Moreover, it requires authentication and cryptographic means to ensure data integrity and confidentiality.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Asset Network/Communication	**Attack** Hijacking/Replay/Suspension/DoS	
Detection • Signature-based Detection [13] • Verification of Safety-Properties [13]	• A precisely calibrated signature effectively identifies abnormal events during software execution. • Ensures that the system does not evolve in unsafe state starting from some initial conditions.	• Does not work when designers and intellectual property providers are not trusted. It cannot handle zero-day attacks and, thus, often used with Anomaly-based techniques leading to a increased resource consumption and time penalty. • It is limited to small scale systems.
Mitigation • Reparameterization [13] • Isolation [11, 13] • Graceful Degradation [13, 15]	• Enables adaptation by switching the configuration parameters of the compromised component to another configuration. • It provides a remedy to enable the system to continue its operation by offsetting the effect of the attack. Also, it prevents loss of functionality. • Prevents a catastrophic failure of the system. It enables a system to continue functioning even after parts of the system have been compromised. It shuts down less critical functions to allocate the resources to more critical functions to maintain availability.	• Decreases the quality of service. • Introduces a time penalty and an increase in required resources (e.g., replica modules that are used to compensate for isolating the affected component of the system). • Causes a degradation in the performance of the operations and services of the system.
Recovery • Relocation/Migration [13, 19] • Software Rejuvenation [11, 19] • Reinitialization [11]	• Maintain system functionality in an operational state as it was before the fault or attack. • Helps avoiding the costs of failures from software degradation, as periodic (graceful) restarts of the software component allow the release and re-allocation of memory, thus, operation in a clean state. • Applied in conditions in which the mitigation is deemed impossible. Restores or pristine resets the system to its initial state.	• May cause an operation of the system in a degraded condition which influences its performance. • Requires shutting the software down and restarting it periodically which causes the software to be unavailable for the duration of the restart. It is often a slow process requiring an extra overhead. • Causes loss of work, and accordingly leads to a waste of resources.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	**Trade-off**	
	Pros	Cons
Endurance • Attack Analysis / Reconstruction (e. g., [57, 58])	• Helps to enhance resilience by systematically and empirically analyzing attacks as well as used technologies (potential entry point, e.g., Bluetooth and WiFi) that interact with the external environment.	• Resource consumption and analysis effort.
Asset Data Storage	**Attack** Unauthorized Read/Manipulation	
Detection • Signature-based Detection [13] • Specification-based Anomaly Detection (e. g., [22])	• A precisely calibrated signature effectively identifies abnormal events during software execution. • Helps detecting anomalies in the system's behavior by reporting the specific deviation that has been observed.	• Does not work when designers and intellectual property providers are not trusted. It cannot handle zero-day attacks and, thus, often used with Anomaly-based techniques leading to a increased resource consumption and time penalty. • Needs of resources for detection and processing of collected information (e.g., costly intelligent sensors). Domain knowledge is required to specify normal behavior. Specifications need to be adapted for each specific vehicle configuration otherwise risk of high false positives or negatives.
Mitigation • Redundancy [11–13, 15, 17–20] • Isolation [11, 13]	• It enables data backup and restore by replicating information and data sources. • It provides a remedy to enable the system to continue its operation by offsetting the effect of the attack. Also, it prevents loss of functionality.	• Requires extra resources for data storage. • Introduces a time penalty and an increase in required resources (e.g., replica modules that are used to compensate for isolating the affected component of the system).
Recovery • Dynamic Deployment of Policies [15]	• Takes the dynamic and changing nature of attacks into account. Deploys different defense policies depending on the attack, for example, it can modify the executed actions while the attack is going on.	• Leads to performance overhead. Requires runtime permissions which may not be present when running normally. Complexity.

Table C.3 – *Continued on next page*

Table C.3 – *Continued from previous page*

Resilience Strategy/ Techniques	Trade-off	
	Pros	Cons
Endurance • Secure Logging (e. g., [54]) • Attack Analysis / Reconstruction (e. g., [57, 58])	• Prevents modifying the logs by using e.g., chained hashes. It enables storing security-related events containing information about e.g., flash operations, external interactions, and power downtime. This information helps to reconstruct events, detect intrusions and identify problems. • Helps to enhance resilience by systematically and empirically analyzing attacks as well as used technologies (potential entry point, e.g., Bluetooth and WiFi) that interact with the external environment.	• Resource consumption and time penalty. Moreover, missing authentication and lack of cryptographic means to ensure data integrity can limit the potential of the logging. • resource consumption and analysis effort.

C.B Proposed Automotive Solutions

In Table C.4 we provide a description of the solutions referred to in Figure C.1. This overview of specific solutions should be considered as a starting point for interested readers and is by no means complete.

Table C.4: Techniques and solutions relevant for the automotive domain.

DETECTION

Pattern	Technique	Solution
Specification-based	Runtime Verification	Heffernan et al. [26] use the automotive functional safety standard ISO 26262 as a guide to derive logical formulae. They demonstrate the feasibility of their proposed runtime verification monitor with an automotive gearbox control system as use case.
	Specification-based Anomaly Detection	Müter et al. [22] describe eight detection sensors that are applicable for the internal network of automotive systems. Six of these sensors are specification-based, e. g., the frequency of specific message types and the range of transmitted values like speed.
Anomaly-Based	Statistical Techniques	Nowdehi et al. [27] propose an IDS that learns about the automotive system by learning from samples of normal traffic without requiring a model definition.
	Machine Learning	Hanselmann et al. [28] propose CANet an unsupervised IDS for the automotive CAN bus. The anomaly score is calculated using the error between the reconstructed signal and the true signal value.

Table C.4 – *Continued on next page*

Table C.4 – *Continued from previous page*

Pattern	Technique	Solution
	Information-theoretic	Müter et al. [29] design an entropy-based IDSs for automotive systems with experimental results using data from a vehicle's CAN-Body network.
	Localization	Cho and Shin [30] present a scheme identifying the attacking ECU based on fingerprinting the voltage measurements on the CAN bus for each ECU. We see great opportunities in the localization of attacks when considering a centralized vehicle architecture combined with virtualisation techniques. This allows us to get detailed performance metrics of virtualized vehicle functions.
Predicting Faults and Attacks	Attack Prediction	Husák et al [31] perform a survey about current attack projection and prediction techniques in cybersecurity.
Redundancy	Diversity Techniques Adaptive Software Diversity	Baudry and Monperrus provide in their survey [71] an overview of different software diversity techniques. Höller et al. [37] introduce an adaptive dynamic software diversity method. The diversification control receives error information from the decision mechanism and randomizes specific parameters during execution. Their experimental use cases demonstrate the dynamic reconfiguration of ASLR parameters, respectively, random memory gaps.
MITIGATION		
Adaptive Response	Model-based Response	Cómbita et al. provide a survey on response and reconfiguration techniques for cyber-physical control systems. Controllers or other systems that can be modelled as a control loop can be, for instance, adjusted to have another module in the feedback loop that compares the actual feedback from the control loop with a simulated/modelled response of what is expected.
Runtime Enforcement	Safety Guard	Wu et al. [44] show how so-called safety guards can be applied to safety-critical Cyber-Physical Systems (CPSs).
Reconfiguration and Reparametrisation	Graceful Degradation	Dagan et al. [38] provide an architectural design on how to extend limp modes so that they can be additionally used in a cyber security context. A safe-mode manager sends out triggering messages that cause the ECUs to transition to a limp mode when cyber-breaches are detected.
		Ishigooka et al. [39] propose a graceful degradation design process for autonomous vehicles with focus on safety.
		Reschkka et al. [40] explore how skills and ability graphs can be used for modelling, on-line monitoring and supporting decision making of driving functions.
	Restructure	Segovia et al. [15] set the focus of their survey on software reflection as mitigation technique for SCADA systems. Software reflection enables the system itself to examine and change its execution behaviour at runtime, which allows, for instance, the system to take actions when an attack is detected. The drawbacks currently seen in software reflection are the performance overhead, the increased execution time and the extended permissions required by software reflection.

Table C.4 – *Continued on next page*

Table C.4 – *Continued from previous page*

Pattern	Technique	Solution
	Dynamic Deployment of Policies	Rubio-Hernan et al. [41] propose an architecture for CPS that combines feedback control loops with programmable networking in order to mitigate attacks by re-routing traffic or applying security rules.
RECOVERY		
Migration	Relocation/ Migration	Jiang et al. [42] propose a hypervisor that meets real-time requirements.
		Other relocation techniques are microservices [72]. Pekka and Mattila [43] propose a service-oriented architecture for real-time CPSs.
	Pre-emptive Migration	Engelmann et al. [73] describe a pre-emptive migration technique which uses a feedback-loop for observing health parameters to detect behaviour indicating a fault. This solution was developed for high performance computing and its applicability for the automotive domain needs to be further investigated.
Checkpointing and Rollback	Software Rejuvenation	Romangnoli et al. [74] describe a method to decide when it is safe to reload the software of a CPS.
ENDURANCE		
Self-*	Continuous Change	Möstl et al. [47] identify in their work the challenges of continuous change and evolution of CPS and propose two frameworks for self-aware systems centring around self-modelling, self-configuration and self-monitoring. The controlling concurrent change (CCC) framework is concerned with how to deal with changes in software components during the lifetime of a CPS. The authors highlight that the well-established V-model currently used is not designed for continuous change and therefore parts of the integration testing and system validation and verification need to be moved to the system itself. The proposed framework includes an automated integration process for new or updated functions that addresses safety, security, availability and real-time requirements. The structure and workflow of the proposed framework is further described using an automotive use case. The second framework concentrates on optimising performance, power consumption and resilience of CPS by using self-organisation and self-awareness techniques.
Verification & Validation	Challenges in V&V	De Lemos [53] discuss research challenges of verification and validation for self-adaptive systems at runtime.
Robustness	Adversarial Attacks on DNN	Yuan et al. [59] give an overview of current adversarial attack and defence techniques for deep learning.
Forensics	Secure Logging	Lee et al. [55] describe T-Box a secure logging solution for automotive systems that makes use of the trusted execution environment in ARM TrustZone.

Table C.4 – *Continued on next page*

Table C.4 – *Continued from previous page*

Pattern	Technique	Solution
		Mansor et al. [56] propose a framework to log vehicle data, such as diagnostic transmission codes, via the mobile phone and store it on a secure cloud storage.
	Attack Analysis / Reconstruction	Nilsson and Larson [57] discuss the requirements for conducting forensic investigations on the in-vehicle network.
		Bortles et al. [58] present which types of data may be retained from current infotainment and telematic systems.

Chapter D

Resilient Shield: Reinforcing the Resilience of Vehicles Against Security Threats

Adapted version that appeared in VTC2021-Spring

K. Strandberg, T. Rosenstatter, R. Jolak, N. Nowdehi, T. Olovsson

Abstract. Vehicles have become complex computer systems with multiple communication interfaces. In the future, vehicles will have even more connections to e.g., infrastructure, pedestrian smartphones, cloud, road-side-units and the Internet. External and physical interfaces, as well as internal communication buses have shown to have potential to be exploited for attack purposes. As a consequence, there is an increase in regulations which demand compliance with vehicle cyber resilience requirements. However, there is currently no clear guidance on how to comply with these regulations from a technical perspective. To address this issue, we have performed a comprehensive threat and risk analysis based on published attacks against vehicles from the past 10 years, from which we further derive necessary security and resilience techniques. The work is done using the *SPMT* methodology where we identify vital vehicle assets, threat actors, their motivations and objectives, and develop a comprehensive *threat model*. Moreover, we develop a comprehensive *attack model* by analyzing the identified threats and attacks. These attacks are filtered and categorized based on attack type, probability, and consequence criteria. Additionally, we perform an exhaustive mapping between asset, attack, threat actor, threat category, and required mitigation mechanism for each attack, resulting in a presentation of a secure and resilient vehicle design. Ultimately, we present the *Resilient Shield* a novel and imperative framework to justify and ensure security and resilience within the automotive domain.

Resilient Shield: Reinforcing the Resilience of Vehicles Against Security Threats

1 Introduction

The complexity of vehicles is increasing. Consequently, vulnerabilities which might be exploited increase as well. Attacks to vehicular systems can be realized: (i) *indirectly* via compromised devices e.g., phones, dongles, or workshop computers connected to vehicle interfaces; (ii) *directly* via physical interfaces e.g., debug ports and the OBD-II connector; and (iii) *remotely* via various malicious sources, such as rogue access points and compromised servers. It has been demonstrated that vehicle cyber attacks e.g., physical attacks [1] and remote attacks [2] are potential threats that have to be taken seriously. As a case in point, Miller and Valasek [3] performed a successful remote attack on a Jeep Cherokee via the Internet taking control of its primary functions by exploiting an open port via a cellular channel, an attack that led to a recall of 1.4 million vehicles. In [4], researchers managed to get remote access to the CAN bus of a BMW by compromising its infotainment system, allowing them to execute arbitrary diagnostic requests. Vulnerabilities in phone applications paired to vehicles have been exploited by adversaries to track vehicles, unlock the doors and to start their ignitions [5–7].

Motivation. Securing a vehicle as an afterthought is cumbersome, considering both the complexity which constantly increases and the existing dependencies on current architectural design. Hence, it is imperative to consider security during the vehicle's complete life cycle from idea to cessation.
There are increased requirements towards ensuring a resilient vehicle design, in a way that a vehicle should be able to withstand various types of cyber attacks, malfunctioning units, and other external disturbances. Consequently, the resilient design should be able to *prevent*, *detect*, and *respond* to cyber attacks, something which is also in line with the UNECE regulation [8] and the upcoming standard for automotive cyber security ISO 21434 [9]. In short, *prevention* is accomplished with security controls, *detection* by identifying faults and attacks, and *response* are mechanisms related to handling the detected anomalies with the ability to restore and maintain operation. However, there is currently no clear guidance how to comply with the aforementioned regulations and standards from a technical perspective. The *start*, *predict*, *mitigate*, and *test* (*SPMT*) is a systematic approach for identification and mitigation of vulnerabilities in vehicles [10]. The aim of *SPMT* is to ultimately enhance the security of vehicles through their entire life cycle. In this paper, we use and extend the *SPMT* methodology to establish an in-depth resilient design model with imperative mitigation mechanisms.

Contributions. By applying the *SPMT* methodology, we performed a comprehensive threat and risk analysis of 52 published attacks against vehicles from the past 10 years. 37 of these attacks were considered significant due to their high risk and were thus further mitigated with imperative security and resilience techniques. In this process, we have developed a *threat model* for securing vehicles by identifying vital vehicle assets and the related potential threat actors, their motivations and objectives. Moreover, we have developed a comprehensive *attack model* created from the analysis of the identified threats and attacks, further filtered and categorized based on attack type and risk criteria related to the probability and consequences of the attack. We present a comprehensive summery of the result from applying the *SPMT* methodology, an exhaustive mapping between asset, attack, threat actor, threat category and resilience mechanism for each attack. Ultimately, we define necessary security and resilience enhancements for vehicles, the *Resilient Shield*, which also validates the effectiveness of the methodology. To the best of our knowledge, our result is both novel and imperative to justify and ensure security and resilience within the automotive domain.

2 Related Work

Good practices for security of smart cars [11], *Cyber security and Resilience of smart cars* [12], and *The Cyber security guidebook for cyber physical vehicle systems, SAE J3061* [13], provide guidelines regarding threat and risk assessment. *EVITA* [14] proposed a method for security, safety, and risk analysis of in-vehicle networks, whereas *HEAVENS* [15] proposed a security model based on security objectives from EVITA and security attributes from Microsoft *STRIDE* [16]. Rosenstatter et al. [17] continue with the result from an analysis such as HEAVENS and map the identified security demands to security mechanisms. However, this mapping focuses only on securing the in-vehicle network.

The *SPMT* methodology builds on existing methods, models and security principles that are applicable to different phases in a vehicle's life cycle. By adapting and incorporating relevant parts suitable for the vehicular domain, a comprehensive security and safety enhancement is achieved. Consequently, the *SPMT* methodology covers the vehicles entire life cycle, something which cannot be achieved with existing methodologies [10]. *SPMT* adopts Microsoft's *STRIDE* categorization [16] which enables a mapping of attacks to a category with associated security attributes. Thus, mitigation mechanisms can be considered for the attribute and consequently mitigate more than one attack. Additionally in *SPMT*, a reduction analysis is performed for critical threats by creating attack trees to connect the vulnerability with the threat, i.e., an attacker wanders from a leaf node (condition) to the root of the tree (attacker objective). Consequently, the closer to the root a countermeasure is placed, the more conditions are mitigated. Moreover, some conditions can be attained by more than one attack, hence a countermeasure can mitigate several attacks.

The REMIND framework [18] for vehicular systems provides a taxonomy for resilience techniques identified from a review of existing work. In this paper we take advantage of previous knowledge and new results by applying the *SPMT* methodology. In the next sections we present the detailed approach followed by the results.

3 Approach

We use the aforementioned *SPMT* model to perform a comprehensive threat modelling and risk assessment of published attacks to further map these threats and attacks to imperative security and resilience mechanisms.

The *SPMT* methodology has 4 phases: *Start*, *Predict*, *Mitigate* and *Test*. In this paper, we perform the first three phases on a Target Of Evaluation (ToE) and analyze security threats and attacks as well as provide mechanisms for the mitigation thereof (see Figure D.1).

- In the *Start Phase*, we address the following questions. *What are the threats requiring a resilient design? What are the entry points to the vehicle? Who are the actors, their motivators, and their objectives?* The outcome of the *Start Phase* is a threat model and high-level goals for the enforcement of security and safety attributes.

- In the *Predict Phase*, we address the following question. *What are the potential attacks?* The outcome of the *Predict Phase* is an *attack model* which contains relevant attacks categorized and filtered according to a stated criteria.

- In the *Mitigate Phase*, we address the following question. *What are the needed mechanisms to ensure a resilient design?* The outcome of the *Mitigate Phase* is a resilient design framework i.e., the *Resilient Shield*, which provides mechanisms and goals for detecting, preventing, and responding to security threats and attacks.

- The *Test Phase* includes the implementation of the mitigation mechanisms followed by an execution of different security tests, such as fuzz, vulnerability, and penetration testing. In this paper, we do not perform the *Test Phase*; however, we plan to test the identified mitigation mechanisms within an industrial context in the future.

In the following sections, we perform and provide the outcomes of the first three phases of the *SPMT* methodology (see Figure D.1) that are used to establish the *Resilient Shield*.

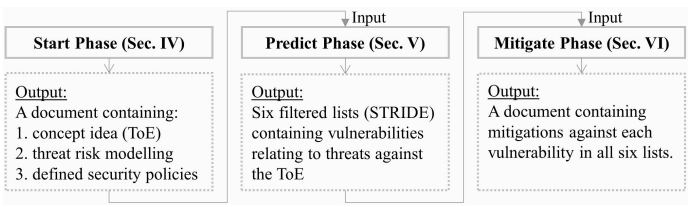


Figure D.1: The first three phases of the SPMT methodology

4 Threat Model

A threat model is created by considering: (i) the target of evaluation (ToE), and (ii) attackers as well as their motivators and objectives. First, our ToE is stated as the complete vehicle provided by the manufacturer, where we propose to include the following assets. As shown in Table D.1, the relevance of these assets is verified by the mapping to attacks.

- ***Internal and external communication:*** *Automotive Bus technologies*, e.g., CAN, FlexRay, LIN, MOST and Ethernet. *Connection interfaces*, e.g., OBD-II, USB, debug ports, Wi-Fi and Bluetooth.
- ***Hardware:*** *ECUs*, e.g., sensor signal processing. *Sensors*, related to speed, position, temperature, airbag and object detection. *Actuators*, translate signals from ECUs into actions, e.g., braking, steering and engine control.
- ***Software in transit, rest or running:*** *Software update systems*, e.g., over-the-air or workshop updates. *Software installed or running* in ECUs.
- ***Data Storage:*** *Sensitive data*, e.g., cryptographic keys, forensics logs and reports.

Second, we propose a simplification of threat actors (i.e., attackers) inspired by the work of Karahasanovic et al. [19] in relation to motivators and objectives.

Actors and Motivators. *The Financial Actor* is driven by financial gain in relation to a company (intellectual property), organization or individual. This actor can be the owner who wants to make unauthorised modifications (e.g., chip tuning) or criminals who install ransomware. *The Foreign Country* is driven by power through cyber warfare, with the intent to disable viable assets within infrastructure (e.g., transportation). *The Cyber Terrorist* is driven by ideological, political or religious objectives. *The Insider* is motivated by retaliation or other personal gains, has knowledge of sensitive information and may plant malicious code into the vehicle. *The Hacktivist* is driven by publicity or adrenaline (i.e., the rush) and can have an agenda for political or social change. *The Script Kiddie* has usually no clear objective, possess limited knowledge and is often using already available tools and scripts. However, the reality is usually a combinations of the mentioned categories and objectives, and actors can be *black hat*, *gray hat*, or *white hat* hackers in relation to society's interpretations of the hackers' intentions. *White hat*, are assumed to be the good guys, *black hats* are the bad guys, and *grey hat* are somewhere in the middle.

Furthermore, in Section 6 we adopt the security and safety attributes used in *SPMT*. These attributes are imperative to uphold to ensure a secure and resilient vehicle. On the other hand, the actors are driven by stated *motivators* (e.g., financial, ideological, publicity) with the goal of compromising these attributes. A discussion and a brainstorming about fulfilment of these attributes is part of the *Start Phase*, however we have chosen to include it in Section 6 to have all considerations for mitigation in one section. Stated assets and actors are applied to Table D.1 and used in the following section.

5 Attack Model

We perform a qualitative risk assessment of published attacks covered in news media and research publications by estimating (i) the probability and (ii) the consequences of the attacks based on the following criteria. As shown in Table D.1, the affected assets, the threat actors and the STRIDE categories for each attack are considered during this assessment.

Attack Probability. The first step in this phase is to define attack probability where the three following estimates should be used:

E1: *Where, when, and in what situation can the attack be carried out?*
E2: *What expertise is required of the attacker?*
E3: *How much time does it take to perform the attack?*

The resulting probability is on a scale of 1 to 3, where 3 indicates that an attack is more probable to take place. The highest value in E1-E3 is chosen.

Attack Consequence. In the second step, the consequences are defined by assessing the effect of the attack on the operational, safety, privacy, and financial aspects. The resulting consequence is on a scale from 1 to 3, where 3 indicates that the consequence is more severe. The highest value is chosen.

Risk Assessment. Once we get the estimates of the attack probability and consequences, we estimate the overall risk by calculating the product of the probability and the consequence, which gives a risk value between 1 and 9 (see Figure D.2). To achieve a realistic balance between the financial cost for mitigation and its related complexity versus the risk and asset value, we consider only the most significant threats. These threats have a risk value of 6 or 9, which is in line with ISO 26262 and ASIL [20] and corresponds to high and critical risk.

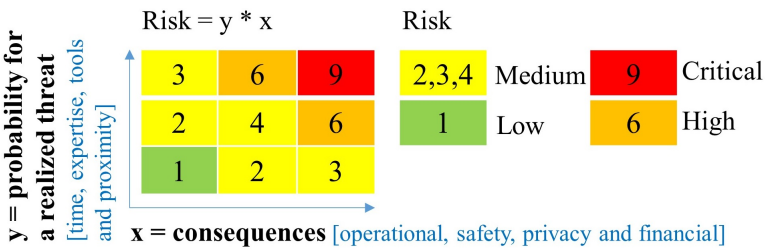


Figure D.2: Adapted table for the risk calculation from the SPMT methodology.

5.1 Disclosed Attacks

To create the *attack model*, we follow the *SPMT* recommendation for search criteria and query scopus[1] and Google scholar for academic work, and common vulnerability databases (NVD, CVE) with keywords related to vehicle, attack and STRIDE categories (e.g., spoofing) or related terms (e.g., mitm). Moreover, we do query the Google search

[1] `https://www.scopus.com/`

engine for media reports on attacks. Next, we classify the attacks according to STRIDE categories, followed by some examples. Attacks are considered and analyzed with respect to probability, consequence and risk within their respective category. Out of a total of 52 published attacks, we have identified 37 high and critical risk attacks which are further considered in this work.

1) Spoofing Attacks - Authenticity, Freshness [5,21–38] The goal of the attacker is to intercept, hijack, manipulate or replay the communication with a potential remote access persistence. *Security flaws in mobile software*, such as demonstrated in the OwnStar attack [5]. OwnStar intercepts communication after the OnStar user opens the application, whereas the OwnStar device gains the user's credentials. Relay attacks, as in compromise of remote keyless entry systems as well as breaking poor authentication mechanisms [21–23]. GNSS spoofing considers broadcasting fake signals over authentic in order to to trick a receiver, with the intention to get a vehicle off course [24]. *In-vehicle protocol spoofing*, can affect safety-critical actuators, such as brake, steering and engine control. Protocols themselves might lack inherent mechanisms for security which makes active attacks possible such as malicious drop, modify, spoof, flood and replay of messages.

2) Tampering Attacks - Integrity [2,4,36,38–41] Vulnerable USB/OBD-II dongles or compromised in-vehicle devices can potentially enable a hacker to control the communication. Devices can be compromised in various ways e.g., vulnerabilities in proprietary authentication mechanisms can enable the right to run sensitive diagnostics commands. Brute-force attacks can be used to retrieve cryptographic keys, with potential to upload exploits to ECUs. Physical tampering of ECUs or other connected devices. Manipulated firmware in current ECUs, such as malicious code injection via firmware update. Replacement of ECUs or new devices to eavesdrop/inject messages or to manipulate software, modify or compromise vehicle functions. Vulnerable connected devices such as OBD and USB dongles can potentially provide remote access to individual cars and vehicle fleets [40]. Moreover, in [2] firmware was extracted and reverse engineered, manipulated and injected directly into ECU firmware facilitating persistent and bridging capabilities for attacks.

3) Repudiation Attacks - Non-repudiation, Freshness An attacker manipulates or removes forensic in-vehicle data, such as GPS coordinates, speed, acceleration and brake patterns, with the intention to hide traces of the attack. Despite our best effort, we did not find attacks which can be clearly mapped to this category; however, this type of attacks will likely be more frequent in the future due to both increased number of attacks and digital forensic investigations.

4) Information Disclosure Attacks - Confidentiality, Privacy [7,38,39,42–45] An attacker may be able to exploit cryptographic keys and consequently decrypt sensitive data by e.g., reverse engineering software with hard-coded keys. Bad routines for handling of replaced unit led to leaked sensitive data such as owners home and work address, calendar and call entries and Wi-Fi passwords [42]. A mobile application for vehicle control contained hard-coded credentials, thus an attacker may be able to retrieve sensitive data remotely by recovering the key from the application [7]. A vulnerability in an OBD-II dongle exposed all transferred

data to the public [43]. Vulnerabilities in automotive bus technologies make various attacks possible, such as sniffing of CAN traffic due to its broadcast transmission and lack of encryption [44].

5) Denial of Service (DoS) Attacks - Availability [34–37, 46–49] Many attacks focus on the in-vehicle network that uses CAN as this technology suffers from fundamental vulnerabilities with respect to security (e.g., broadcast communication, lack of encryption/authentication). Other attacks range from sending an indefinite amount of data to ECUs to make them unresponsive or crash, exploiting error handling mechanisms, or flooding the network with high priority messages in order to block lower priority messages. A vulnerability in the Bluetooth functionality supported unrestricted pairing without a PIN, thus enabled the potential for sending remote CAN commands affecting safety-critical assets [48]. The Bus-off attack made ECUs unresponsive or crash [49]. Murvay et al. [47] managed to disable FlexRay nodes by exploitation of the bus guardian, power saving functionality and by causing loss of synchronization.

6) Elevation of Privilege Attacks - Authorization [3, 7, 36, 38, 39, 41, 50–52] In [36] two Bluetooth vulnerabilities allowed remote code execution with root privileges. Moreover, manipulation of the firmware of the infotainment unit enabled injection of arbitrary CAN messages. In [50], they were able to release the airbag by message injection due to a vulnerable authentication mechanism. Lack of authentication in the NissanConnect app allowed to retrieve personal data by entering an URL with the vehicle identification number [52]. The outcome of this phase is applied to Table D.1 and used in the next phase in the following section.

6 Resilient Shield

In this section we present the *Resilient Shield* which consists of high-level security goals emphasizing the overall design requirements resulting from an analysis of the threat model (Section 4). We further provide in Section 6.2 detailed directives for fulfilling the high-level security goals for resilient vehicles which are based on these goals and the *attack model* (Section 5). Table D.1 summarizes the *Resilient Shield*. We list automotive assets, associate them with high risk attacks, potential threat actors and STRIDE threat categories, and link these to suitable security and resilience techniques to show how *Resilient Shield* can be used to mitigate these attacks.

6.1 High-level Security Goals (SGs)

The following high-level goals are the result of an analysis of the *threat model* detailed in Section 4. Each SG is associated with the relevant safety and security attributes they enforce.

SG.1 Secure Communication. *Integrity*, *authenticity* and, in specific cases, *confidentiality* need to be ensured for communication. *Integrity* and *authenticity* allow to verify the origin of the message and protect the message from being altered during transmission. *Confidentiality* can be achieved through encryption of the message to

prevent unauthorized read access. *Freshness*, e.g., via counters or timestamps, can be used to mitigate replay attacks.

SG.2 Readiness. *Availability* to authorized entities under normal circumstances as well as disturbances. Even if an adversary tries to disrupt the information flow, the *integrity* and *availability* of correct information needs to be guaranteed.

SG.3 Separation of Duties is needed to limit access to resources for *authorized* entities only. *Authorization* should be combined with the principle of *least privilege* to limit the number of entities having access to a resource to the minimum.

SG.4 Secure Software Techniques need to provide security features to ensure that the executed software has not been modified by an unauthorized entity (*authenticity*) and that the software does not contain disclosed vulnerabilities.

SG.5 Separation/Segmentation on an architectural or process level is necessary in order to limit access and reduce the severity in case of an intrusion (*availability*). *Isolation* techniques, e.g., process isolation, should be considered where possible.

SG.6 Attack Detection and Mitigation is of utmost importance to enable the system to react and ideally prevent further damage to the system.

SG.7 State Awareness should be ensured with the ability to switch between various operational states, thus providing *reliability* and *maintainability*.

SG.8 Forensics is necessary for post analysis of detected malicious events and accordingly updating access control policies and other preventive measures.

Physical security, such as vehicle locks, alarm system, and protecting infrastructure server rooms should be considered. Components must be extensively tested against requirements separately and when integrated into the vehicle, such as stated in the *SPMT Test Phase*. *SPMT* suggests to use both a qualitative and quantitative assessment; however, we focus on the qualitative assessment as the aim of *Resilient Shield* is to guide the resilient design of automotive systems. Moreover, a reduction analysis of attack trees is suggested to find commonalities in countermeasures; however this is not considered and is thus left as future work.

6.2 Detailed Directives

In this section, we list detailed techniques and patterns that contribute to the security and resilience of automotive systems based on the identified security goals, *threat* and *attack model* presented in this paper. First, we incorporate the identified patterns from the REMIND framework [18] in *Resilient Shield* and further extend them with security techniques to provide a comprehensive collection of both, security and resilience techniques for automotive systems. Second, we further discuss the security aspects of the identified resilience techniques. Next, we detail these techniques.

Authentication. Message authentication can be achieved through Message Authentication Codes (MACs) or signatures which ensure that the message: (i) is created by the claimed source and (ii) has not been altered during transmission. The authentication of devices can verify that the hardware, e.g., the head unit or a diagnostic device, is legit.

Encryption. Encryption of data ensures the protection of intellectual property, makes it more difficult to reverse engineer software, protects cryptographic material and the privacy of users and forensics data.

Redundancy/Diversity. A voting mechanism is used when comparing the output of two or more redundant systems or software functions. Redundancy increases the resilience against anomalies; however, from a security perspective it must be ensured that the voting process cannot be exploited by an attacker to perform DoS or spoofing attacks.

Access Control. Gateways with firewall capabilities allow filtering of messages between different networks in the vehicle. In addition, host-based firewalls on the ECUs can limit the exposure of open communication ports. Securing physical debug ports is vital to protect against unauthorized exploitation. Access control to resources such as files, computation, and diagnostic commands can be provided by the operating system or by e.g., challenge-response authentication.

Runtime Enforcement. Runtime verification is combined with reactive measures when safety properties are violated [18, 53].

Secure Storage. Cryptographic material needs to be protected against unauthorized modifications and read access. Data can be either stored encrypted in the regular file system or in a protected memory partition.

Secure Boot. A validation of the authenticity and integrity of the firmware to be loaded during the boot process [54].

Secure Programming. Secure programming guidelines such as MISRA C [55] are important to avoid common programming errors. Additionally, trusted execution environments may be necessary for isolating and securing applications.

Secure Software Update. The ability to update software is not only a necessity to improve and extend functionality, it is also essential for security, e.g., to mitigate vulnerabilities. In addition, the update process itself needs to be secure [56], during the distribution and installation process.

Verification & Validation. The *Test Phase* in *SPMT* focuses on the need for security testing and verification of each asset by doing fuzz, vulnerability and penetration testing. In addition to security testing, the verification and validation of functionality and safety is required [10, 18].

Separation. Architectural separation can be achieved through physical separation into smaller networks or through virtualization techniques allowing to allocate resources to specific functions or systems.

Specification-based Detection. Knowledge about abnormal behavior is used to detect anomalies and attempts to exploit known vulnerabilities. It also requires domain knowledge and needs to be updated regularly [18, 57].

Anomaly-based Detection. Is based on defining normal behavior and deviations trigger alerts and has the potential to detect unknown attacks. Anomaly-based detection can be categorized in statistical, information-theoretic, machine learning and localization techniques [18, 57].

Prediction of Faults/Attacks. Predicting the next step or the ultimate goal of an ongoing attack.

Adaptive Response. The function response may be temporarily adapted, e.g., through a model, while under attack [18].

Reconfiguration. Graceful degradation can be used to limit the impact of an attack when preventive measures failed.

Migration. The ability to migrate services to other nodes in order to maintain system functions when under attack [18].

Checkpoint & Rollback. Used to recover the system to a desired state. The state needs to be secured, e.g., through secure logging, to defend against possible attacks that aim at modifying a saved system state [18].

Rollforward Actions. Upon detecting an anomaly or error the system transitions back to the state immediately before the event happened. Similarly to rollback it needs to be ensured that this mechanism cannot be exploited [18].

Self-X. The system needs to be aware of its state and able to switch to other states when anomalies occur [18, 58].

Robustness. Employed mechanisms and functions need to be robust against anomalies [18].

Forensics. Secure logging is used to record events, e.g., detection of an ongoing attack, use of specific services or diagnostics. In addition, events with non-repudiation claims can be used as evidence of a crime.

Table D.1 presents the *Resilient Shield*. Assets with high or critical risk threats are associated with appropriate security and resilience techniques demonstrating the ability of *Resilient Shield* to defend against these threats. For example, hacktivists and insiders are the main threat actors for *communication:external:debugport*, such as JTAG, and needs to be protected with authentication mechanisms combined with access control or, if not possible otherwise, with physical protection (e.g., deactivation).

7 Conclusion

We have performed a comprehensive threat and risk analysis of published attacks against vehicles and derived imperative security and resilience mechanisms by applying the *SPMT* methodology. A *threat model* with vital vehicle assets and related potential threat actors, their motivations and objectives was developed. By an extensive analysis of threats and attacks, further filtered and categorized based on attack type, probability and consequence criteria, an *attack model* was developed based on the remaining high risk attacks. Based on the developed models, a comprehensive mapping between asset, attack, threat actor, threat category, and defense mechanisms was performed for all attacks and is presented in Table D.1. Table D.1 summarizes the outcomes by applying *SPMT*, i.e. the *Resilient Shield*, a novel framework both justifying and defining imperative security and resilient mechanisms needed in a modern vehicle. Consequently, the *Resilient Shield* can be used as a vital baseline for protection against common security threats and attacks.

Table D.1: Resilient Shield. A mapping from automotive assets to identified attacks, potential threat actor ultimately to appropriate security and resilience techniques, and Security Goals (SGs).

Assets targeted by attacks with high or critical risk.
ToE category:subcategory reference

■ Resilience patterns identified in REMIND [18]

Potential Threat Actors
Financial Actor (FA)
Foreign Country (FC)
Cyber Terrorist (CT)
Insider (IN)
Hacktivist (HA)
Script Kiddie (SK)

STRIDE categories
(S)poofing
(T)ampering
(R)epudiation
(I)nformation Disclosure
(D)enial of service
(E)levation of privilege

			(SG.1,8) Authentication	(SG.1) Encryption	(SG.2) Redundancy/Diversity ■	(SG.3) Access Control	(SG.3) Runtime Enforcement ■	(SG.4,8) Secure Storage	(SG.4) Secure Boot	(SG.4) Secure Programming	(SG.4) Secure Software Update	(SG.4) Verification & Validation ■	(SG.5) Separation ■	(SG.6) Specification-based Detection ■	(SG.6) Anomaly-based Detection ■	(SG.6) Prediction of Faults/Attacks ■	(SG.6) Adaptive Response ■
Hardware																	
sensor:camera [34, 35]	FC, CT, HA	S, D			•												•
sensor:GNSS [24, 26, 29, 30, 32]	FC, CT, HA	S	•		•										•		•
sensor:lidar [28, 34]	FC, CT, HA	S, D			•												•
sensor:ultrasonic [35]	FC, CT, HA	S, D			•												•
Communication																	
internal:can [40, 44, 46, 47, 49]	FA, FC, CT, IN, HA	S, T, I, D	•	•	•	•	•						•	•	•	•	
internal:flexray [37]	FA, FC, CT, HA	S, D	•			•	•						•	•	•		
external:bluetooth [4, 36]	FC, CT, HA	S, T, D, E	•			•							•				
external:usb [4]	FC, CT, HA	S, T, E	•			•							•				
external:keyfob [22, 23]	HA, SK	S	•			•									•		
external:wifi [5, 33]	HA, SK	S, I	•	•		•				•			•				
external:cellular [3, 4, 41, 45, 51, 52]	FC, CT, HA, SK	S, T, I, D, E	•			•							•				
external:obdII [7, 27, 31, 38, 40, 43, 46, 48]	CT, HA	S, T, I, D, E	•			•	•						•	•	•		
external:debugport [3, 41]	HA, IN	I, E	•			•											
Software																	
running:state [25]	FC, CT, HA	S, D				•				•				•	•		
running:firmware [3–5, 33, 36, 39, 41, 45, 51, 52]	FC, CT, HA	S, T, E				•				•	•	•	•	•	•		
instorage:update [4, 36, 41]	HA, SK	S, T, E	•	•		•		•	•		•		•	•	•	•	•
instorage:weakcrypto [21, 50, 52]	FC, CT, HA, SK	S, E	•							•							
Data Storage																	
crypto:certificates [41]	FC, CT, HA	I		•		•		•	•								
hw:replaced [42]	HA, SK	I	•	•		•											

We believe our work is imperative for facilitating and guiding the design of resilient automotive systems; however, it still remains to be seen how large the coverage is in relation to future attacks. Moreover, testing and validation of the *Resilient Shield* within an industrial context is left as a future work.

Acknowledgments. This research was supported by the CyReV project (2019-03071) funded by VINNOVA, the Swedish Governmental Agency for Innovation Systems.

Bibliography

[1] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Security Symposium*. San Francisco, 2011.

[2] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 447–462.

[3] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, 2015.

[4] Tencent Keen Security Lab, "Experimental Security Assessment of BMW Cars: A Summary Report," https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf, 2018, accessed: 2020-09-11.

[5] S. Kamkar, "Drive it like you hacked it: New attacks and tools to wirelessly steal cars," *Presentation at DEFCON*, vol. 23, 2015.

[6] CVE Details, "Security vunerabilities bluelink," https://www.cvedetails.com/vulnerability-list/vendor_id-16402/product_id-37376/Hyundaiusa-BlueLink.html, accessed: 2020-09-11.

[7] CVE List, "CVE-2019-9493," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9493, accessed: 2020-09-11.

[8] UNECE, "Draft recommendation on cyber security of the task force on cyber security and over-the-air issues of UNECE wp.29 GRVA," UNECE, Tech. Rep., 2018.

[9] "ISO/SAE 21434 Road Vehicles – Cybersecurity Engineering," International Organization for Standardization (ISO), Standard, 2020.

[10] K. Strandberg, T. Olovsson, and E. Jonsson, "Securing the connected car: A security-enhancement methodology," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 56–65, 2018.

[11] "Good practices for security of smart cars," ENISA, Tech. Rep., 2019.

[12] "Cyber security and resilience of smart cars," ENISA, Tech. Rep., 2016.

[13] "SAE J3061: Cybersecurity guidebook for cyber-physical vehicle systems," SAE International, Standard, 2016.

[14] EVITA, "EVITA deliverables," https://www.evita-project.org/deliverables.html, accessed: 2020-09-11.

[15] M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson, "A risk assessment framework for automotive embedded systems," *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, 2016.

[16] Microsoft, "The STRIDE threat model," https://msdn.microsoft.com/en-us/library/ee823878.aspx, 2005, accessed: 2020-09-11.

[17] T. Rosenstatter and T. Olovsson, "Towards a standardized mapping from automotive security levels to security mechanisms," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 1501–1507.

[18] T. Rosenstatter, K. Strandberg, R. Jolak, R. Scandariato, and T. Olovsson, "REMIND: A framework for the resilient design of automotive systems," *IEEE Secure Development*, 2020, in press.

[19] A. Karahasanovic, P. Kleberger, and M. Almgren, "Adapting threat modeling methods for the automotive industry," *15th ESCAR, Berlin*, 2017.

[20] "ISO 26262:2011 Road Vehicles – Functional Safety," International Organization for Standardization (ISO), Standard, 2011.

[21] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlidès, "Lock it and still lose it—on the (in) security of automotive remote keyless entry systems," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016.

[22] A. Greenberg, "Just a pair of these $11 radio gadgets can steal a car," https://www.wired.com/2017/04/just-pair-11-radio-gadgets-can-steal-car/, accessed: 2020-09-11.

[23] A. Francillon, B. Danev, and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. ETH Zürich, Department of Computer Science, 2011.

[24] M. L. Psiaki and T. E. Humphreys, "GNSS spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, 2016.

[25] K. Iehira, H. Inoue, and K. Ishida, "Spoofing attack using bus-off attacks against a specific ECU of the CAN bus," in *15th IEEE Consumer Communications & Networking Conference (CCNC)*, 2018, pp. 1–4.

[26] Q. Meng, L. Hsu, B. Xu, X. Luo, and A. El-Mowafy, "A GPS spoofing generator using an open sourced vector tracking-based receiver," *Sensors*, vol. 19, no. 18, p. 3993, 2019.

[27] CVE List, "CVE-2019-12797," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-12797, accessed: 2020-09-11.

[28] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park *et al.*, "Adversarial sensor attack on LiDAR-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2267–2281.

[29] Cyware Hacker News, "Seven car manufacturers hit by GPS spoofing attacks," https://cyware.com/news/seven-car-manufacturers-hit-by-gps-spoofing-attacks-146701c4, accessed: 2020-09-11.

[30] Help Net Security, "Research shows Tesla Model 3 and Model S are vulnerable to GPS spoofing attacks," https://www.helpnetsecurity.com/2019/06/19/tesla-gps-spoofing-attacks/, accessed: 2020-09-11.

[31] CVE, "CVE-2018-11478," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-11478, accessed: 2020-09-11.

[32] D. Schmidt, K. Radke, S. Camtepe, E. Foo, and M. Ren, "A survey and analysis of the GNSS spoofing threat and countermeasures," *ACM Comput. Surv.*, vol. 48, no. 4, May 2016. [Online]. Available: https://doi.org/10.1145/2897166

[33] Pen Test Partners, "Hacking the Mitsubishi Outlander PHEV hybrid," https://www.pentestpartners.com/security-blog/hacking-the-mitsubishi-outlander-phev-hybrid-suv/, accessed: 2020-09-11.

[34] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, p. 2015, 2015.

[35] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *DEF CON*, vol. 24, no. 8, p. 109, 2016.

[36] Tencent Keen Security Lab, "Experimental security assessment on lexus cars," https://keenlab.tencent.com/en/2020/03/30/Tencent-Keen-Security-Lab-Experimental-Security-Assessment-on-Lexus-Cars/, 2020, accessed: 2020-09-15.

[37] P. Murvay and B. Groza, "Practical security exploits of the FlexRay in-vehicle communication protocol," *International Conference on Risks and Security of Internet and Systems*, pp. 172–187, 2019.

[38] Argus Cyber Security, "A remote attack on the Bosch Drivelog connector dongle," https://argus-sec.com/remote-attack-bosch-drivelog-connector-dongle/, accessed: 2020-09-11.

[39] CVE List, "CVE-2016-9337," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-9337, accessed: 2020-09-11.

[40] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, 2015.

[41] M. Yan, J. Li, and G. Harpak, "Security Research on Mercedes-Benz: From Hardware to Car Control," https://i.blackhat.com/USA-20/Thursday/us-20-Yan-Security-Research-On-Mercedes-Benz-From-Hardware-To-Car-Control.pdf, 2020, accessed: 2020-09-15.

[42] G. H. Ruffo, "Tesla Data Leak: Old Components With Personal Info Find Their Way On eBay," https://insideevs.com/news/419525/tesla-data-leak-personal-info-ebay/, accessed: 2020-09-11.

[43] CVE List, "CVE-2018-11477," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-11477, accessed: 2020-09-11.

[44] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures challenges and future directions," *IEEE Network*, 2017.

[45] J. C. Norte, "Hacking industrial vehicles from the internet," http://jcarlosnorte.com/security/2016/03/06/hacking-tachographs-from-the-internets.html, accessed: 2020-09-11.

[46] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2017.

[47] P. Murvay and B. Groza, "DoS attacks on controller area networks by fault injections from the software layer," *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017.

[48] CVE List, "CVE-2016-2354," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-2354, accessed: 2020-09-11.

[49] K. Cho and K. Shin, "Error handling of in-vehicle networks makes them vulnerable," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[50] J. Dürrwang, J. Braun, M. Rumez, and R. Kriesten, "Security evaluation of an airbag-ECU by reusing threat modeling artefacts," in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2017, pp. 37–43.

[51] T. Brewster, "BMW updates kills bug in 2.2 million cars that left doors wide open to hackers," https://www.forbes.com/sites/thomasbrewster/2015/02/02/bmw-door-hacking/, 2015, accessed: 2020-09-11.

[52] T. Hunt, "Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs," https://www.troyhunt.com/controlling-vehicle-features-of-nissan/, 2016, accessed: 2020-09-11.

[53] M. Wu, H. Zeng, C. Wang, and H. Yu, "INVITED: Safety guard: Runtime enforcement for safety-critical cyber-physical systems," in *54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[54] S. Sanwald, L. Kaneti, M. Stöttinger, and M. Böhner, "Secure boot revisited," *17th escar Europe*, 2019.

[55] *MISRA C: Guidelines for the Use of the C Language in Critical Systems 2012*. Motor Industry Research Association, 2013.

[56] T. Karthik, A. Brown, S. Awwad, D. McCoy, R. Bielawski *et al.*, "Uptane: Securing software updates for automobiles," *14th ESCAR Europe*, 2016.

[57] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805 – 822, 1999.

[58] C. G. Rieger, D. I. Gertman, and M. A. McQueen, "Resilient control systems: Next generation design research," in *2009 2nd Conference on Human System Interactions*, 2009, pp. 632–636.

Part III

Design and Evaluation of Security and Resilience Techniques

Chapter E

Extending AUTOSAR's Counter-based Solution for Freshness of Authenticated Messages in Vehicles

Adapted version that appeared in PRDC 2019

T. Rosenstatter, C. Sandberg, T. Olovsson

Abstract. Nowadays vehicles have an internal network consisting of more than 100 microcontrollers, so-called Electronic Control Units (ECUs), which control core functionalities, active safety, diagnostics, comfort and infotainment. The Controller Area Network (CAN) bus is one of the most widespread bus technologies in use, and thus is a primary target for attackers. AUTOSAR, an open system platform for vehicles, introduced in version 4.3 SecOC Profile 3, a counter-based solution to provide freshness in authenticated messages to protect the system against replay attacks. In this paper, we analyse and assess this method regarding safety constraints and usability, and discuss design considerations when implementing such a system. Furthermore, we propose a novel security profile addressing the identified deficiencies which allows faster resynchronisation when only truncated counter values are transmitted. Finally, we evaluate our solution in an experimental setup in regard to communication overhead and time to synchronise the freshness counter.

Extending AUTOSAR's Counter-based Solution for Freshness of Authenticated Messages in Vehicles

1 Introduction

The internal network of modern vehicles consists of more than 100 Electronic Control Units (ECUs) that communicate via different network technologies, such as Controller Area Network (CAN) and Ethernet. CAN is a relatively old technology developed by Robert Bosch GmbH in 1983 and later standardised in ISO 11898 [1]. Back then, securing the CAN bus was not an issue, as listening and transmitting on the bus required special equipment and physical access to the vehicle. This assumption changed when vehicles became connected to the outside world, e. g., the internet, mobile phones, and other vehicles.

Miller and Valasek demonstrated a remote exploitation back in 2015 [2], where they were able to remotely control a vehicle over the Internet. Given this, and other successful attacks in more recent years underlines the need for security in the automotive domain, especially, a secure transmission of data.

AUTOSAR, a system architecture developed by a consortium of vehicle OEMs and suppliers, defines in *Specification of Secure Onboard Communication v4.4.0* [3] three so-called SecOC Profiles, which provide message authentication to ensure that messages were sent by said origin and have not been altered during transmission. Messages are authenticated by calculating the corresponding Message Authentication Code (MAC) and sending it along with the clear text. SecOC Profile 2 verifies the authenticity of messages without an additional counter or timestamp, whereas SecOC Profile 1 and 3 use a counter-based Freshness Value (FV) in order to prevent replay attacks. The difference between the latter two is that SecOC Profile 3 includes a mechanism to synchronise the FV across senders and receivers. Such a mechanism to synchronise the FV is required when only a truncated FV is sent along with the data, for instance, due to the limited payload size and network technology used in automotive networks.

In this paper, we analyse and assess AUTOSAR's SecOC Profile 3 in detail, discuss design aspects and limitations of counter-based freshness solutions, and lastly propose a new profile that is independent of the underlying network architecture and addresses the identified issues. Additionally, we evaluate our method in terms of communication overhead and time to synchronise the FV.

2 AUTOSAR SecOC Profile 3 (JASPAR)

The AUTOSAR *Secure Onboard Communication Specification* [3] introduces SecOC Profile 3, also named JASPAR, a method that guarantees freshness and provides message authentication. Profile 3 applies CMAC/AES-128 [4] for message authentication and is used in combination with a master/slave synchronisation of a freshness counter as explained in [3, Annex A]. Profile 3 also introduces an entity called Freshness Value Manager (FVM), which is responsible for maintaining the current FV and synchronising the value between senders and receivers. The FVM can either run centrally on a dedicated ECU or decentralised in each sending ECU. A method for a fast synchronisation of the FV between the different entities is necessary, as the senders and receivers normally only store a portion of the FV in Non-Volatile Memory (NVM). In addition, a watchdog timer reset or other unexpected situations may occur where an ECU restarts and consequently fails to successfully verify incoming Interaction Layer Protocol Data Units (I-PDUs) due to the fact that the FV has already changed.

In the following sections we describe the FV, the structure of the I-PDUs and continue with giving details under what circumstances Profile 3 is able to correctly verify the authenticity of I-PDUs with truncated FVs even when I-PDUs are lost.

2.1 Freshness Value and I-PDU Format

Profile 3 describes two configurations for sending data:

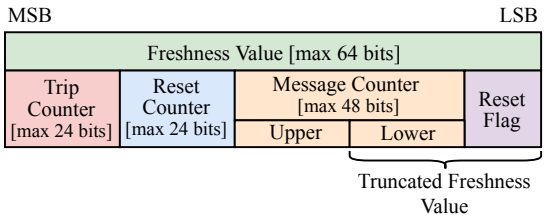
(1) sending the authenticated data, named *Authentic I-PDU*, and the authentication information, namely the *Cryptographic I-PDU*, separately; or

(2) sending the application data and the authentication information in one I-PDU, a *Secured I-PDU*, which is only possible if the data to be authenticated is short.

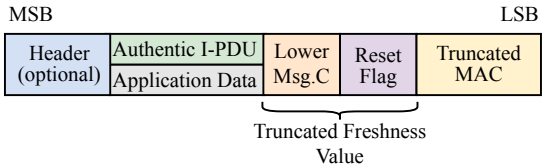
A truncated FV and MAC, which is further called authenticator, may be sent for each I-PDU since sending these values in full length significantly increases the network load. CAN, for instance, is already highly utilised and requires a truncation as the maximum payload size is 64 bits.

Freshness Value. The FV has a maximum size of 64 bits and consists of three sub-counters (their maximum size is given in braces): trip counter (24 bits), reset counter (24 bits) and message counter (48 bits), as shown in Figure E.1a. Note that the length of the sub-counters must be adjusted individually and may not exceed the maximum size of the FV (64 bits). The trip counter is increased in units of trips, the reset counter is incremented periodically defined by a static parameter *ResetCycle* and the message counter is increased per message/I-PDU being sent. The reset flag shown in Figure E.1a contains the n least significant bits of the reset counter.

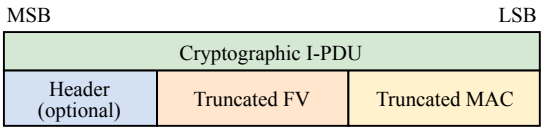
Authentic I-PDU. This I-PDU shown in Figure E.1b contains the data that is authenticated by the sender. Depending on the configuration it is sent apart from the authenticator and FV, or as part of the Secured I-PDU.



(a) Structure of the locally stored Freshness Value.



(b) Structure of the Secured I-PDU, i. e., containing both, data and authentication information.



(c) Structure of the Cryptographic I-PDU, i. e., containing only the authentication information.

(d) Structure of the Synchronisation Message.

Figure E.1: Structure of the FV and I-PDUs [3, p.138,145].

Secured I-PDU. Is generated when the Authentic I-PDU, the authenticator and the FV are combined in one I-PDU. It contains an optional header, the data to be transmitted, the truncated FV and the authenticator, as shown in Figure E.1b. AUTOSAR recommends to transmit the lower 4 bits of the FV and the upper 28 bits of the MAC. Note that the authenticator contains the MAC of the plain text (the message) and the full length FV even in cases when only a truncated FV is transmitted.

Cryptographic I-PDU. Figure E.1c gives details about this I-PDU which is sent along with the Authentic I-PDU and thus allows, in case of CAN, more bits of the FV and MAC to be transmitted.

Synchronisation Message. Also named TripResetSyncMsg, is sent periodically by the FVM when either the trip counter increases or a new *ResetCycle* starts, for

instance every second. The synchronisation message, see Figure E.1d, contains the trip and reset counter in their full length as well as the corresponding authenticator. The message counter is set to zero when a synchronisation message is received. A synchronisation message is also sent at startup or after a reset of the FVM. In such a case the trip counter stored in NVM gets incremented and the reset counter is reset to 0.

2.2 Reconstruction of the Freshness Value

When transmitting only a truncated FV along with each PDU, situations can occur in which the receiver will not be able to correctly verify the authenticity of the received I-PDUs due to a mismatch of the current FV, for example after an ECU reset by a watchdog timer or woken up after a long sleep. Profile 3 describes in [3, Figure 6-7] how to reconstruct the counter values when only truncated FVs are transmitted and the internally stored counters do not result in a successful verification of the Authentic I-PDU. *(1)* The receiver then tries to verify the I-PDU with its internally stored FV+1. *(2)* If this verification fails, it updates the FV with the truncated value from the received I-PDU, e. g., lower 2 bits of the reset and message counters. *(3)* The internally stored value consisting of the trip counter, reset counter and upper part of the message counter is incremented by 1 and the verification is retried. An internal counter, called *attempts*, is incremented with every repetition of *(3)* until the allowed maximum (parameter R can be configured) is reached and the I-PDU will be dropped.

The rest of this section describes two situations where it is not possible to reconstruct the correct counter value resulting in the ECU being unable to verify the authenticity of the I-PDU. As the truncated FV contains a few bits of the reset and message counters, the receiver is able to correctly verify the authenticity of I-PDUs even if it has missed some I-PDUs. We assume in the rest of this analysis that the *ResetCycle* is harmonised with the message counter, meaning that a synchronisation message is always sent when the message counter overflows. The abbreviations used in the following analysis are as follows:

R	The maximum number of verification retries.
m	The length of the message counter.
m_l	The length of the lower part of the message counter.
r_f	The length of the reset flag.
C_{last}	The message counter value of the last successfully verified message.
T	The period in which messages are sent.

2.2.1 Reconstruction of message counter fails

The first situation in which the receiver is not able to correctly verify the authenticity of an I-PDU is when it misses too many I-PDUs to be able to correctly reconstruct the actual message counter assuming that the reset counter did not change. In this situation, it is possible to reconstruct the message counter as long as no more than $2^{m_l} + R \cdot 2^{m_l}$ PDUs are missed. Overall, the SecOC module is able to recover the correct FV as long as it receives an I-PDU with a message counter C in the range:

$$C_{last} < C \leq min\{C_{last} + 2^{m_l} + R \cdot 2^{m_l}, 2^m - 1\} \tag{E.1}$$

The total waiting time for a synchronisation message since the last successfully verified I-PDU is

$$(2^m - 1 - C_{last}) \cdot T \tag{E.2}$$

Example. We analyse a system with an 8 bit message counter $m = 8bits$; $m_l = 2bits$; $C_{last} = 0$; $R = 2$. AUTOSAR recommends a truncated FV of length 4, which corresponds to $m_l = 2bits$ and $r_f = 2bits$ in our example. We chose to illustrate the worst case scenario, when the last correctly verified counter value is 0, for illustration purposes, however, the identified interval applies for all counter values. In this scenario the receiver can reconstruct the correct FV as long as it receives one of the next consecutive 12 I-PDUs ($2^2 + 2 \cdot 2^2$, see Eqn. E.1). If it misses more than 12 PDUs, respectively the ECU sleeps longer than 240 ms with a message frequency of 50Hz, the module is unable to recover and has to wait in total $2^8 - 1 - 0 = 255$ PDUs (see Eqn. E.2) since the last successfully verified PDU to receive the next synchronisation message. Figure E.2 illustrates this example when the SecOC module receives only the truncated FV and MAC with each PDU. The current counter values in each step are aligned with their corresponding counter and presented in binary format.

In case that the SecOC module receives a truncated FV (see Figure E.1b) of $0b0000$ and the last received message counter was 0, it is still able to correctly verify the authenticity of the I-PDU when the real message counter value is $0b00001100$. Note that the module will drop the I-PDU as soon as the MAC verification fails and the number of maximum attempts R is reached. This example shows that it is not possible (with $m_l = 2bits$) to correctly verify I-PDUs with a message counter $\geqslant 0b00001101$ as more than two attempts to reconstruct the correct message counter ($R > 2$) would be needed.

2.2.2 Reconstruction of the reset counter fails

The second situation occurs when the SecOC module has been inactive for a longer period and consequently missed at least one synchronisation message. In this situation, the module is able to recover the correct FV as long as the message counter is in the range described in Section 2.2.1 and as long as the truncated reset counter, the reset flag, can be used to correctly restore the FV. For example, a 2-bit reset flag can be used to correct $2^2 - 1$ reset counter increments. Overall, the interval in which it is possible to correctly reconstruct the FV with changing reset counter is illustrated in Figure E.3.

3 Design Considerations and Limitations

In addition to the time span ECUs may be out of sync, non-functioning and waiting for a synchronisation message, we have identified other potential challenges when introducing SecOC Profile 3.

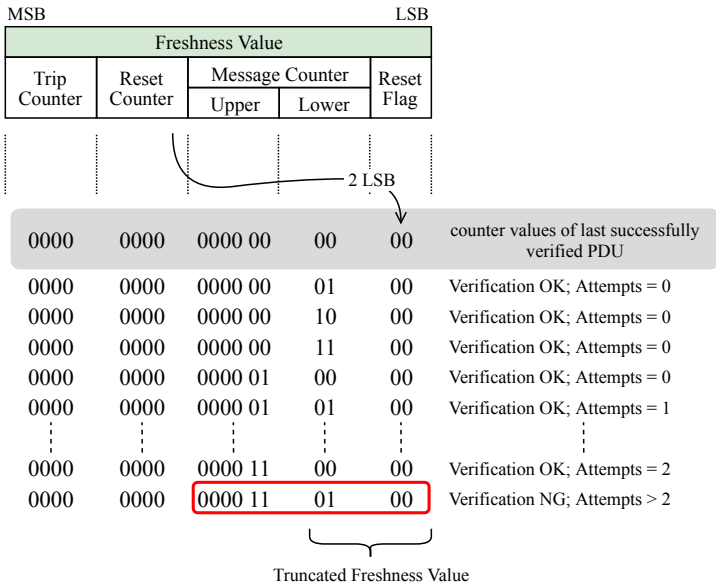


Figure E.2: Example illustrating when an ECU is not able to correctly verify an I-PDU.

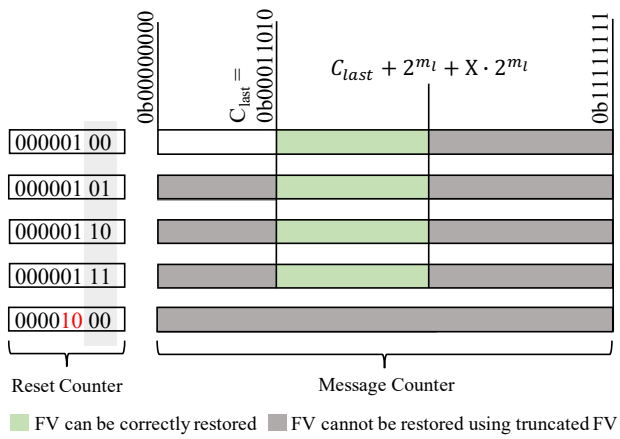


Figure E.3: Example showing the cases when the ECU is able to correctly verify the I-PDU when the reset counter increases.

Freshness Value Manager (FVM). The FVM may be implemented decentralised, meaning that each sending ECU has its own instance of the FVM, or centralised, where there is one FVM for all senders and receivers.

The trip and reset counters are shared between all senders and receivers when

applying the centralised approach, however, the trip counter is the only counter that has to be stored in NVM of the FVM. Storing and accessing only a small portion of the FV in NVM is faster and increases also the lifetime, which is expected to be at least 10 years, as less space in memory pages has to be written. The reset and message counters are stored in volatile memory where the message counter is the only counter maintained individually per I-PDU/message type by the sender and receivers.

CAN uses so-called CAN-IDs to distinguish between different message types. For this reason, the synchronisation message of each FVM requires its own distinct CAN-ID. Thus, a centralised approach also reduces the number of required CAN-IDs due to having one synchronisation message for all senders and receivers. The decentralised approach on the other hand requires each sender to maintain the current trip counter in their NVM as this counter is independent of the counters used by other senders.

Having a centralised FVM might bring in challenges regarding the propagation of synchronisation messages, as gateways cause additional delays. This, however, depends on the chosen placement of the FVMs, e. g., having one FVM per network segment or one for the entire internal network of the vehicle.

A centralised FVM also introduces the FVM as a single point of failure and thus increases the complexity drastically when combined with safety-critical functions. As an example, a safety-critical system, which is classified as ASIL D, in combination with Profile 3 would require a redundant FVM with dissimilar software and hardware redundancy according to ISO 26262 [5], the functional safety standard for road vehicles. From this perspective, a decentralised FVM is desirable for safety-critical functions.

Single-I-PDU configuration. Sending data and its authenticator in a single I-PDU is faster compared to sending them separately for the reason that the data has to be kept in Random Access Memory (RAM) until the second PDU containing the corresponding authenticator is received. Heavy duty vehicles have to comply to SAE J1939 [6], which may require to send the authenticator as a separate frame for the reason that this standard defines the content of certain CAN frames to provide interoperability between a variety of equipment for heavy duty vehicles. Thus, sending data and authenticator separately allows increased security for core functions developed by the vehicle OEM while third-party modules are still able to receive the predefined messages defined in [6]. Sending two frames also allows the transmission of longer truncated values when considering the maximum payload size of 64 bits in CAN.

Complex Structure of the FV. Profile 3 introduces a FV consisting of three counters (see Figure E.1a). The trip counter is incremented in units of trips, which requires a global understanding of the unit as it strongly depends on the function of the ECU. AUTOSAR specifies that the trip counter shall be incremented when the ECU running the FVM starts, on wakeup, on reset and when the power status changes from off to on. ECUs in a heavy duty vehicle might be active for several days, e. g., interior light control, whereas others will boot when the ignition is switched on. Thus, it may happen that ECUs miss the increment of the trip counter and need to wait for the next synchronisation message. Moreover, the use of the reset counter is

solely to indicate that a new ResetCycle has started. For this reason, we do not see an advantage of having the reset counter separated from the message counter.

Reset Counter Overflow. There are no means to increase the trip counter when the reset counter overflows. Instead, a synchronisation message with the maximum value of the reset counter will be sent, meaning that the freshness property of transmitted messages no longer applies.

Determining the Reset Cycle. The parameter *ResetCycle* is used to define the frequency of synchronisation messages. The *ResetCycle* should be harmonised with the maximum value of the message counter in order to achieve the highest utilisation of the counter space as synchronisation messages reset the message counter.

A centralised FVM introduces additional challenges, as the *ResetCycle* is defined globally for all I-PDUs in the vehicle. In this case the counter space as well as the periodicity of the synchronisation message cannot be efficiently adjusted for individual I-PDU types.

Periodicity of Synchronisation Messages. Since synchronisation messages are broadcast periodically as defined by the parameter *ResetCycle*, receiving ECUs will be in a non-functional state until a synchronisation message is received when they are out of sync. There is no way for a receiver to notify the FVM that it needs synchronisation of the FV.

Having means to request synchronisation messages is necessary in order to provide a fast resynchronisation, such as when starting a vehicle by turning on the ignition or when an ECU encounters a watchdog timer reset. Most vehicles, including heavy duty vehicles, have different modes of operation, such as run, accessory, parked, living and crank. *RUN* indicates that the vehicle is driving and fully operational, *LIVING* that ECUs related to the driving functionality are shut off and only comfort ECUs, such as interior lighting and infotainment, are active. Given these dynamics of the vehicle modes, it is necessary to have a fast recovery when an ECU loses track of the current FV.

From a safety perspective it is desirable to have a deterministic, maximum specified interval for synchronisation messages. In addition, having fast resynchronisation of non-safety-critical functions may be important for comfort functions to provide a "premium" feeling when driving a vehicle.

4 Proposed SecOC Profile 4

We address the identified issues in our proposed profile which offers:

- a faster synchronisation of the FV.
- alignment with ISO 26262.
- less bandwidth usage due to the reduced number of control messages needed to synchronise the FV.
- a simplified structure of the FV.

We propose two modes of operation: *Mode 1* similar to Profile 3, however, a new structure for the FV is used; *Mode 2* a more efficient and faster approach that allows

receiving ECUs to request synchronisation messages on demand instead of having to wait for the next periodic synchronisation message. Figure E.4 provides an overview of the different configurations, modes and message types used. *Configuration 1* and *2* describe in which format the data and authenticator are exchanged, either in one I-PDU or separately. We propose using the same structure as defined in AUTOSAR for the following I-PDUs: Authentic, Secured and Cryptographic I-PDU. When truncating the FV we use the same approach as Profile 3 described in [3, Figure 6-7].

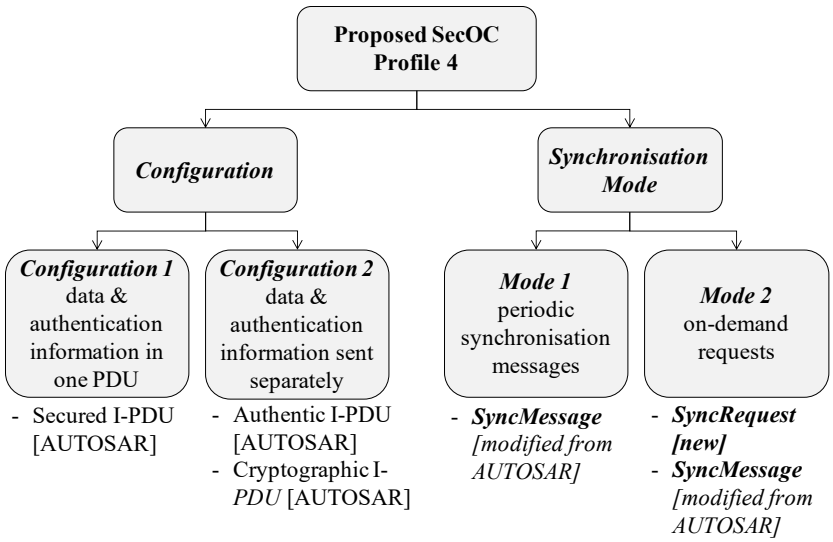


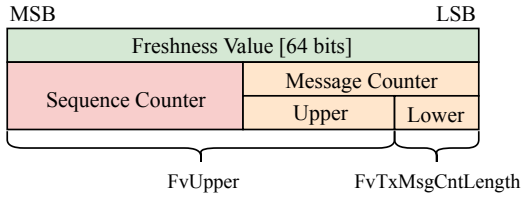
Figure E.4: Overview of configuration and mode options including the involved messages.

We describe the structure of the FV and the new and modified messages in Section 4.1. Sections 4.2 and 4.3 describe the different modes of synchronisation followed by a description of default parameters and recommended values in Section 4.4.

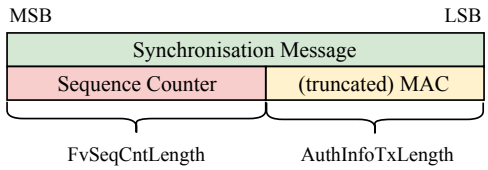
4.1 Freshness Value and I-PDU Format

The structure of the FV and I-PDUs in our proposed method is described in this section and shown in Figure E.5.

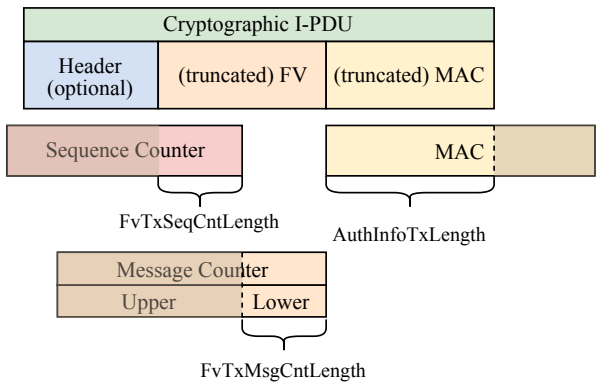
Freshness Value. Figure E.5a depicts the FV, which is reduced to having only two sub-counters, i. e., a sequence counter and a message counter, which is, similar to Profile 3, split into an upper and lower part. The sequence counter is maintained by the FVM and thus needs to be stored in NVM. The scope of the sequence counter can be defined to be either one per FVM or one counter per message type. Moreover, the sequence counter may be increased due to a SyncMessage being sent for the reason that the FVM has been restarted, encountered an error, received an internal trigger to increase the sequence counter, or has received a SyncRequest.



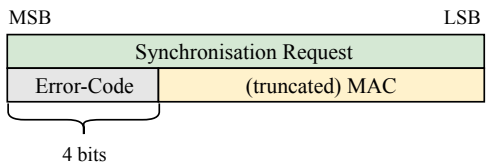
(a) Structure of Freshness Value (*Mode 1, 2*).



(b) Synchronisation Message sent either periodically (*Mode 1*) or by request (*Mode 2*).



(c) Cryptographic I-PDU sent along with Authentic I-PDU (*Configuration 2*).



(d) Synchronisation Request (*Mode 2*).

Figure E.5: Structure of the proposed FV and I-PDUs.

Authentic and Secured I-PDU. The structure is similar to Profile 3 (see Figure E.1b). Configuration 1 sends the Authentic I-PDU separately or combined with the authenticator when Configuration 2 is enabled.

Cryptographic I-PDU. Is sent when Configuration 2 is enabled. Figure E.5c explains how to combine the sequence and message counter when truncated FVs are transmitted.

SyncMessage. The synchronisation message contains the sequence counter and its corresponding MAC. As this message is used to synchronise the FV between senders and receivers, there is, similar to SecOC Profile 3, no possibility to provide freshness for this message. The nodes, however, must verify that the received sequence counter is larger than sequence counters previously used to prevent replays of old messages from an adversary.

SyncRequest. Receivers may send on-demand requests for synchronisation in case they have no knowledge about the current sequence counter or cannot successfully verify the received I-PDUs. The Error-Code depicted in Figure E.5d can be used to signal the FVM why the ECU demands a new SyncMessage, for instance due to consecutive verification fails or a reboot.

4.2 Sending Periodic SyncMessages (Mode 1)

This configuration is similar to Profile 3, the parameter *ResetCycle* defines the frequency in which SyncMessages are broadcast and shall be chosen by considering the maximum time an ECU is allowed to be out of sync without degrading its functionality. We recommend using this configuration only when necessary due to its impact on the bandwidth of the underlying network.

4.3 On-Demand Request for Synchronisation (Mode 2)

This mode enables the FVM to send SyncMessages when SyncRequests are received. In situations, such as when multiple receivers wake up, i. e., changing to an active state or when an ECU being stuck in a bootloop, may occur and result in several SyncMessages being sent within a short time frame. Given these circumstances, the FVM is required to handle multiple SyncRequests within a short period of time. A dynamic parameter, such as *SyncMessageSuspend*, can be used to define the time during which SyncRequests are ignored after a SyncMessage has already been sent. Therefore situations when a SyncMessage has already been sent by the FVM but not yet received by the ECU, which sends another SyncRequest when it realises that it is out of sync, are covered as well.

SyncRequests are, similar to SyncMessages, vulnerable to replay attacks as these two messages are used to regain synchronisation of the FV. Restricting the number of SyncRequests in combination with a monotonically increasing FV, however, provide also security against attackers who inject previously recorded SyncRequests in order to overflow the sequence counter and thus force the FVM to reuse counter values. Considering the scenario of a 28 bit sequence counter and a limit of 2 SyncRequests per second would take an attacker 4.2 years until the FV is repeated.

A CAN specific implementation of the SyncRequest may make use of so-called Remote Frames (RFs) [1]. RFs can be used to request data by sending a frame with the same CAN-ID as the requested data, where only the Remote Transmission Request

(RTR) bit changes. This specific solution for CAN has the advantage that no additional CAN identifier for the SyncRequest is needed and thus does not additionally exhaust the pool of available CAN-IDs. The CAN-ID is also used to prioritise the frames – the lowest CAN-ID has the highest priority (see arbitration in [1]). Coupling the priority with the CAN-ID demands further consideration when choosing an ID, as SyncRequests are event-triggered and not periodic.

4.4 Recommendations and Default Values

We recommend combining both modes, i. e., sending periodic SyncMessages and allowing ECUs to request them (Mode 1+2), for messages that need be authenticated. By enabling receivers to request SyncMessages, a larger *ResetCycle* and thus longer message counter, which in turn reduces the load on the network, can be used. The *ResetCycle* not only depends on the chosen mode, it also strongly depends on the requirements and for how long an ECU is allowed to be unable to correctly verify the authenticity of I-PDUs due to being out of sync.

The size of the FV and the authenticator depend on several factors, such as computational and storage limitations of senders and receivers, the bus technology, and current network load. Unless compliance to standards, i. e., SAE J1939 for heavy duty vehicles, is required, it has to be decided whether the data of an existing CAN frame can be split in two frames to create space for the authenticator and the truncated FV (configuration 1 in Figure E.4) or if a second frame containing only the authenticator and the truncated FV (configuration 2) should be used.

Table E.1 lists our recommended parameter values, described in Figures E.4 and E.5, for mode 1 and 2. These values are based on the AUTOSAR SecOC Profile 3 recommendations and should be considered as a base for further adaptation depending on the requirements. The number of additional attempts for reconstructing the FV depends strongly on the message frequency of the authenticated data as well as the computational power of the ECU.

5 Experiments and Evaluation

The experimental setup to validate the functionality of our proposed solution is shown in Figure E.6 and consists of three nodes, one sender and two receivers, which were implemented on Freescale MPC 5646C microcontrollers with a compliant AUTOSAR software using the AUTOSAR 4.3 crypto stack. The FVM is executed on the sending ECU, as we believe that a decentralised approach is more realistic to be deployed in a production environment. Furthermore, we chose to send the authentic data separated from the authentication information (i. e., use configuration 2) as this setting provides both backward compatibility and compatibility with standards specifying the frame content.

We have analysed the time a resynchronisation takes when only mode 1 or both modes were activated using the parameters shown in Table E.2. These parameters where chosen to highlight the differences between the two modes, other parameters such as the length of the authenticator, are not relevant for the following analysis.

Table E.1: Recommended parameters

Parameter	**Mode 1**	**Mode 1+2**
ResetCycle	5 Hz	1 Hz
	Freshness Value	
Attempts	2	
FvLength	64 bits	
FvUpper	FvLength – FvMsgCntLower	
FvSeqCntLength	28 bits	
FvMsgCntLength	FvLength – FvSeqCntLength	
	Configuration 1	**Configuration 2**
AuthInfoTxLength	28 bits	44 bits
FvTxSeqCntLength	2 bits	4 bits
FvTxMsgCntLength	2 bits	16 bits

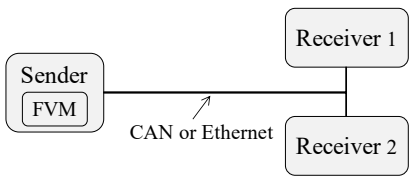
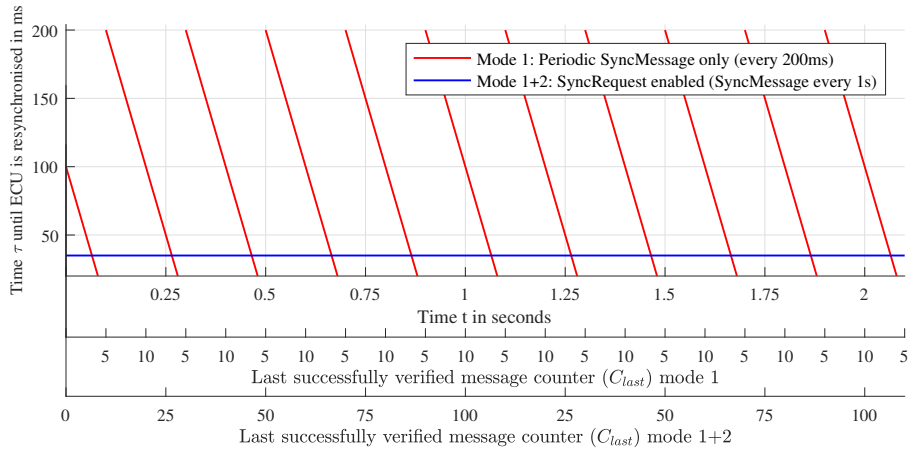


Figure E.6: Experimental setup where sender and receivers either communicate via CAN or Ethernet.

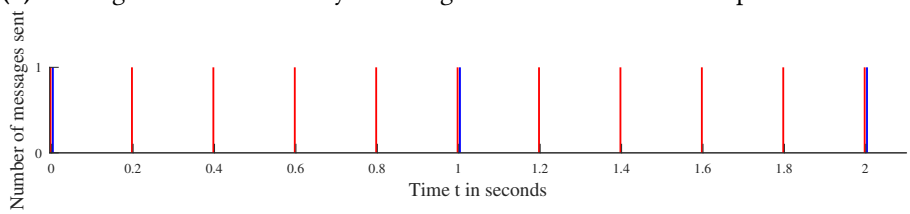
Figure E.7 shows the calculated times from a scenario where the receiver gets interrupted for 90 ms, e. g., due to a watchdog timer reset or an erroneous event. It shows the time τ, which is the time measured between receiving the first I-PDU after the interruption and the successful resynchronisation of the ECU. Figure E.7a illustrates the behaviour of periodic SyncMessages; τ decreases with the increasing counter value C_{last}, as the next periodic synchronisation message approaches. The large τ

Table E.2: Parameters used for comparing mode 1 and mode 1+2

Parameter	**Mode 1**	**Mode 1+2**
ResetCycle	5 Hz	1 Hz
Attempts	0	0
FvMsgCntLength	2 bits	2 bits
max. message counter value	10	100



(a) Waiting time for the next SyncMessage when the ECU is interrupted for 90ms.



(b) SyncMessages being sent.

Figure E.7: Analysis of the waiting time when an ECU is interrupted for 90 ms.

from $C_{last} \geq 5$ is due to the interruption of 90ms, which causes the receiver to miss the periodic SyncMessage. The blue line shows that enabling requests for synchronisation messages, has a constant time τ since the generation and transmission of the SyncRequest and SyncMessage have a fixed delay. In a real setting, there will be a variation of τ depending on the bus load and priority of the messages. However, it is reasonable to assume that even without using very high priorities, these messages should always be possible to be transmitted within 20 ms. In addition, as shown in Figure E.7b, enabling SyncRequests provides not only a faster resynchronisation of the FV, it also makes it possible to reduce the number of periodic SyncMessages.

5.1 CAN Test Bed

In this setting, we first analyse the processing times and delays of the sending ECU respectively the FVM. The sender is configured to send a SyncMessage every 200 ms and to accept SyncRequests. These settings were chosen to highlight that the proposed method is also faster when combined with shorter periods for sending SyncMessages. Figure E.8 illustrates the frequencies of different messages transmitted on the CAN bus generating in total a bus load of 14%, where messages in cyan colour are generated background traffic. The messages with IDs 8300311 and 8300313 are the Authentic and Cryptographic I-PDU destined for Receiver 1 with a message

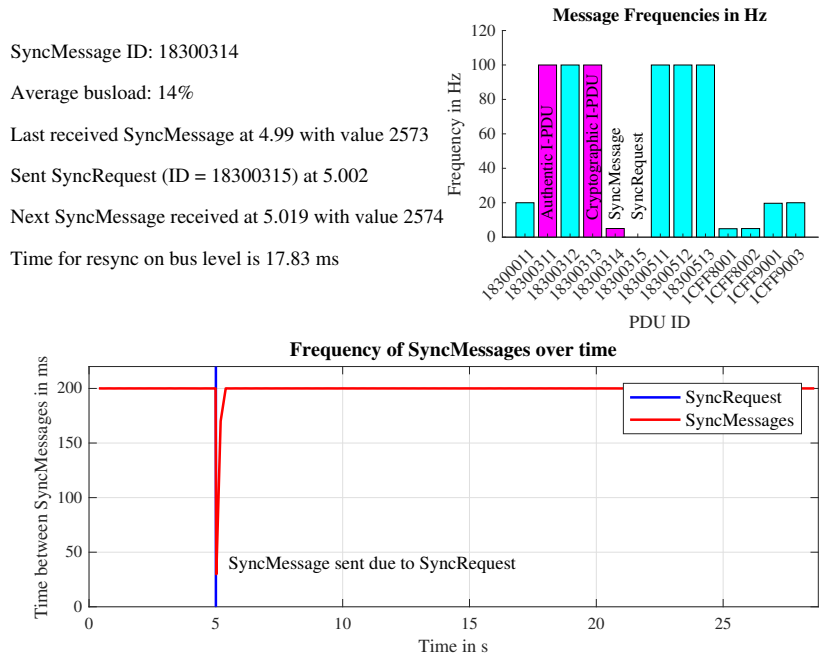


Figure E.8: Results when the receiving ECU gets out of sync and sends a SyncRequest.

frequency of 100 Hz. SyncMessage and SyncRequest were chosen to have a lower priority (in CAN the lowest ID has the highest priority) than the actual secured message to show that our approach also works well even when messages with higher priorities are transmitted on the network. We force the receiving ECU to get out of sync by triggering the sending ECU/FVM to immediately increase the sequence counter by 100.

In our measurements it takes the sender 17.8 ms from receiving the SyncRequest at second 5.00 to sending a SyncMessage, which corresponds to the spike shown in the chart presenting the time between SyncMessages in Figure E.8.

Overall, it takes the FVM, respectively the sending ECU, in average 18.1 ms to transmit the SyncRequest, process the request and transmit the corresponding SyncMessage. Receiver 1 needed in average 25 ms from recognising that it is out of sync until having a synchronised FV again. Comparing this fast resynchronisation to sending only periodic SyncMessages as shown in Figure E.7a highlights that not just the waiting time can be greatly reduced, but also the period in which SyncMessages are sent can be extended to reduce the bus load of the network. To achieve the same average for resynchronisation as we have achieved by enabling SyncRequests would require sending SyncMessages with a frequency of at least 20 Hz.

Other factors such as the bus load and number of receiving ECUs have an impact on the time a resynchronisation takes when on-demand SyncRequests are enabled. In most cases it can be assumed that SyncRequests and SyncMessages can be sent

within the next 20 ms on CAN even with high bus loads when assigning the priorities properly. Measures to prevent flooding of SyncMessages by many ECUs sending SyncRequests close after each other may impact the response time of the FVM as well. Such measures can be implemented on the FVM and may be to limit the number of SyncRequests within one SyncMessage period (ResetCycle) or to only send one SyncMessage within a certain time frame.

A case that might require a special handling of SyncRequests is the KeyOn event and other events where many ECUs are expected to startup simultaneously. In such scenarios a fast ECU might request and receive a SyncMessage while others are still booting. One approach is to allow more SyncRequests within one ResetCycle during startup to allow a fast synchronisation. Another approach to cope with many ECUs being out of sync is to temporarily set a faster period for sending SyncMessages and increase the ResetCycle later on. For instance, for the KeyOn scenario one may set the ResetCycle to 50 ms for two seconds and afterwards increase it to the regular period of 1 second.

5.2 Ethernet Test Bed

The test bed with the nodes communicating over Ethernet with each other was used to validate that our implementation, specifically the use of our structure of the FV, works on Ethernet as well. We chose to send the complete FV along with every message for the reason that the requirement of having a highly limited bandwidth, as in CAN, does not apply. Thus, there is no need for sending synchronisation messages or requests. We successfully validated the functionality of our proposed approach using the structure of the UDP payload shown in Figure E.9.

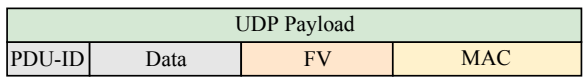


Figure E.9: Structure of the UDP payload sent over Ethernet (Secured I-PDU).

6 Related Work

Zou et al. [7] identify the challenges of time and counter-based solutions for CAN. One of the identified challenges for counter-based solutions is the need for ECUs to be able to request synchronisation messages, however, the authors do not provide more details.

Gürgens and Zelle [8] present a CAN specific hardware-based solution to provide counter-based freshness. This method uses one counter per CAN bus where the counter gets incremented by 1 when a message is sent. The authors also mention that their proposed solution cannot be implemented in currently available ECUs as the CAN transceivers require additional functionality. Limiting the scope of the counter/FV per CAN network segment additionally increases also the delay through

gateways due to the verification and subsequent generation of a new MAC using the counter of the network the message is forwarded to.

VulCAN [9] proposes a trusted computing design for message authentication including software component attestation, but leaves the resynchronisation of the FV to the underlying protocol being used.

Existing CAN authentication solutions based on industrial criteria have been evaluated by Nowdehi et al. [10]. According to Nowdehi et al., VatiCAN [11] fulfils the requirements of cost effectiveness, backward compatibility, and repair and maintenance. VatiCAN provides freshness using a nonce, but it requires to be synchronised periodically (i. e., the authors suggest every 50 ms).

7 Conclusion

Freshness is an important security property that ensures that authenticated messages have not been replayed by a malicious entity. Counter-based solutions are especially interesting in the automotive domain, as sensors and other Electronic Control Units (ECUs) rarely have a globally synchronised clock. AUTOSAR proposed two counter-based solutions, one using a single counter and another one providing additionally a master/slave synchronisation of the Freshness Value (FV).

In this paper, we focus on the second solution presented by AUTOSAR, namely SecOC Profile 3 or JASPAR. We first provide a detailed analysis of Profile 3 which shows in which situations the recovery mechanism of the FV succeeds when only a truncated FV is transmitted. Second, we study the limitations, safety impact and other design considerations when implementing a counter-based solution to provide freshness for signals in the in-vehicle network. Third, we propose an extension of Profile 3 that allows a faster synchronisation of the FV in case senders or receivers have lost track of the current FV due to, for instance, an unexpected reset, internal state change or waking up from sleep. We further evaluate our proposed solution on two test beds each communicating via CAN bus and Ethernet. The experiment shows that our proposed solution is significantly faster in synchronising ECUs. Moreover, the number of necessary control messages used to synchronise is reduced notably and will have a positive effect on the bus load.

Acknowledgements. This research was supported by the HoliSec project (2015-06894) funded by VINNOVA, the Swedish Governmental Agency for Innovation Systems.

Bibliography

[1] "ISO 11898-1:2015 Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling," International Organization for Standardization (ISO), Standard, 2015.

[2] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[3] AUTOSAR 4.4.0, *Specification of Secure Onboard Communication*, 2018. [Online]. Available: https://www.autosar.org/fileadmin/Releases_TEMP/Classic_Platform_4.4.0/Communication.zip

[4] T. Iwata, J. Song, J. Lee, and R. Poovendran, "The AES-CMAC Algorithm," RFC 4493, Jun. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4493.txt

[5] "ISO 26262:2011 Road Vehicles – Functional Safety," International Organization for Standardization (ISO), Standard, 2011.

[6] "Serial Control and Communications Heavy Duty Vehicle Network - Top Level Document," SAE International, Tech. Rep., 08 2013. [Online]. Available: http://doi.org/10.4271/J1939_201308

[7] Q. Zou, W. K. Chan, K. C. Gui, Q. Chen, K. Scheibert, L. Heidt, and E. Seow, "The study of secure CAN communication for automotive applications," in *WCX 17: SAE World Congress Experience*. SAE International, mar 2017. [Online]. Available: https://doi.org/10.4271/2017-01-1658

[8] S. Gürgens and D. Zelle, "A hardware based solution for freshness of secure onboard communication in vehicles," in *Computer Security*, S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, A. Antón, S. Gritzalis, J. Mylopoulos, and C. Kalloniatis, Eds. Cham: Springer International Publishing, 2019, pp. 53–68.

[9] J. Van Bulck, J. T. Mühlberg, and F. Piessens, "VulCAN: Efficient component authentication and software isolation for automotive control networks," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC 2017. New York, NY, USA: ACM, 2017, pp. 225–237. [Online]. Available: http://doi.acm.org/10.1145/3134600.3134623

[10] N. Nowdehi, A. Lautenbach, and T. Olovsson, "In-vehicle CAN message authentication: An evaluation based on industrial criteria," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, 2017.

[11] S. Nürnberger and C. Rossow, "vatiCAN – vetted, authenticated CAN bus," in *Cryptographic Hardware and Embedded Systems – CHES 2016*, B. Gierlichs and A. Y. Poschmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 106–124.

Chapter F

Team Halmstad Approach to Cooperative Driving in the Grand Cooperative Driving Challenge 2016

Adapted version that appeared in T-ITS 2018

M. Aramrattana, J. Detournay, C. Englund, V. Fridmodig, O. Uddman Jansson, T. Larsson, W. Mostowski, V. Díez Rodríguez, T. Rosenstatter, G. Shahanoor

Abstract. This paper is an experience report of team Halmstad from the participation in a competition organised by the i-GAME project, the Grand Cooperative Driving Challenge 2016. The competition was held in Helmond, the Netherlands, during the last weekend of May 2016. We give an overview of our car's control and communication system that was developed for the competition following the requirements and specifications of the i-GAME project. In particular, we describe our implementation of cooperative adaptive cruise control, our solution to the communication and logging requirements, as well as the high level decision-making support. For the actual competition we did not manage to completely reach all of the goals set out by the organisers as well as ourselves. However, this did not prevent us from outperforming the competition. Moreover, the competition allowed us to collect data for further evaluation of our solutions to cooperative driving. Thus, we discuss what we believe were the strong points of our system, and discuss post-competition evaluation of the developments that were not fully integrated into our system during competition time.

Team Halmstad Approach to Cooperative Driving in the Grand Cooperative Driving Challenge 2016

1 Introduction

In the European Union (EU), road transportation stood for 75% of the inland goods transportation in 2014. In 2013, passenger cars accounted for 83% of the inland passenger transport. Combustion of fuel used for transport produced 23% of the CO_2 gas emissions in EU during 2014, and road transport accounted for 25.8% of the European energy consumption in 2013 [1]. To reduce the environmental impact and reach the 2°C ceiling target, EU and several governments have clear goals on how to handle this societal challenge and specifically to reduce these emissions, e.g., Sweden has an aim to have a fossil fuel independent transportation sector with the target to reduce fossil fuel consumption by 80% until 2030 [2].

Traffic safety is another large societal challenge. In EU, 28,000 fatalities were reported in 2012 [1] and traffic accidents are among the ten most common causes of deaths according to the World Health Organization[1]. Worldwide, traffic accidents is the most common cause of death for young people aged 10–24 [3].

Urbanisation and demographical changes are other challenges for the future transportation system. Denser city population and more elderly drivers will further strain the transportation system. Moreover, limited space and high building cost make it difficult to expand roads. Automated and cooperative vehicles are one possible strategy to overcome all these challenges. The roads can be utilised more efficiently, and the environmental impact can be reduced by being able to drive with shorter inter-vehicular distance, which reduces air resistance and consequently lower the energy consumption. Furthermore, by unburdening the drivers with more automated functions the traffic safety will improve.

1.1 Related work within cooperative and automated driving

Early projects within this field of research are, e.g., PROMETHEUS (Program for European Traffic with Highest Efficiency and Unprecedented Safety) [4] that was running between 1988–1995, with the vision to create intelligent vehicles as a part of an overall intelligent road traffic system. Other European projects are, e.g., the CVIS[2] and the SAFESPOT[3] projects. The California PATH (Partners for Advanced Transportation Technology) was initiated in 1986 and is still on-going. PATH pioneered platooning and demonstrated the first Automated Highway System (AHS) in

[1] http://www.who.int.

[2] https://cordis.europa.eu/project/id/027293.

[3] http://www.safespot-eu.org.

1994 with a four-car platoon featuring automated longitudinal control [5].

The Safe Road Trains for the Environment (SARTRE) project, running in 2009–2012, was co-funded by the European Commission within the 7th Framework Program. SARTRE aimed at developing strategies and technologies to allow platooning within regular public highways to create environmental, safety, and comfort benefits. The SARTRE project demonstrated the benefits of platooning with reported fuel savings of up to 20% for the members of the platoon [6]. Another related FP7 project is AutoNet 2030 [7] which investigated how the heterogeneous fleet of vehicles could cooperate to increase safety and fluidity within traffic. Consequently, the project studied both what information should be exchanged between the different road users and how the road users should be organised (centralised or distributed).

In the same way as levels of automation have been defined, for example by SAE[4], three dimensions of cooperation in ITS and driving has been proposed [8]. These dimensions are (1) individual, local, or global scope; (2) operational, tactical, or strategical task; and (3) two, three, or more actors. Platooning is an example that requires cooperative behaviours in the task (operational, tactical, and strategical), scope (individual, local, and global) and number of actors (2 or more). Cooperative adaptive cruise control and operation concepts [9] are important parts of platooning but it does not cover, e.g., the lane change manoeuvres that are needed.

To perform cooperative behaviour related to the formation, joining, or leaving a platoon on a highway, different types of cooperation, coordination, and agreement protocols have been proposed and evaluated in simulated scenarios [10–12]. There is also work related to the higher level issues to find out which vehicles can gain on forming a platoon, taking into account, e.g., that they have similar goals at the more strategic task level [13]. To handle advanced cooperative behaviour several messages need to be exchanged within limited time. This is a serious problem with Inter Vehicle Communication (IVC) [14] and especially in highly congested traffic. The authors of [15] address these problems by proposing and evaluating a slotted beaconing protocol as a time organised alternative to the ETSI ITS-G5 proposed protocols Decentralized Congestion Control [16] and Dynamic Beaconing [17].

1.2 Grand Cooperative Driving Challenge

In 2011, the first Grand Cooperative Driving Challenge (GCDC 2011) [18] was arranged. The goal of GCDC 2011 was to accelerate the development, integration, demonstration, and deployment of cooperative mobility. In GCDC 2011, two scenarios were demonstrated, one highway and one urban scenario. In the urban scenario a traffic light-controlled intersection was used to coordinate two platoons in the same lane that were instructed to join after each other. In the highway scenario it was demonstrated how shock waves, that are common on highways, can be attenuated by using cooperative adaptive cruise control supported by V2V communication, i.e. making use of position and speed information similar to the content of a Cooperative Awareness Message (CAM). Furthermore, a dedicated GCDC cooperative interaction protocol was designed to enable execution of the intersection scenario.

[4] http://standards.sae.org/j3016_201609/.

GCDC 2016 organised by the i-GAME (Interoperable GCDC AutoMation Experience) project defined new competition scenarios, which besides adding lateral manoeuvres, introduce the most important additional challenge compared to GCDC 2011. That is, to use cooperative lane change messages to handle joint operations among pairs of vehicles driving in two adjacent platoons. It also includes coordinating if and when to alter distances between vehicles and when to change lane in a cooperative manner. The iCLCM (i-GAME Cooperative Lane Change Message) and protocol needed for this were developed by i-GAME before the challenge. Consequently, the challenge was also an essential field test of this approach.

The main challenge in GCDC, both 2011 and 2016, is the multi-vendor approach, where vehicles of different size and brand, developed by different teams at different locations, are going to collaborate and perform cooperative manoeuvres on a real highway at a considerably high speed (80 km/h).

1.3 Contribution

This paper summarises the system developed by team Halmstad for the GCDC 2016 competition, which builds on the experiences gained in GCDC 2011 [19], and elaborates on distributed vehicle coordination. The competition consists of three scenarios: (a) merging of two platoons on a highway; (b) cooperative intersection crossing; and (c) a demonstration of an intelligent emergency vehicle warning application. All scenarios are enabled by distributed negotiation where vehicles communicate to coordinate with each other. A brief description of the GCDC scenarios is given in Section 2, a more detailed description of the scenarios and GCDC can be found in [20]. In this paper, we describe our implementation of cooperative adaptive cruise control, our solution to the communication and logging requirements, as well as the high level decision making support. Due to space restrictions and the intended character of this paper, the solutions are described with a varying level of detail. Consequently, our communication and trust system solutions are discussed in greater detail in two accompanying publications [21, 22].

One of the approaches developing our system for the competition was to use, where possible, cost efficient hardware. The paper describes the associated challenges, most vividly in communication, and how they were addressed. In this context, the competition was a source of real data, which was used to further develop and evaluate our ideas and solutions.

1.4 Paper Organisation

The rest of the paper is organised as follows; Section 2 gives an overview of the GCDC 2016 scenarios. Section 3 describes the experimental setup of the vehicle and the architecture of the developed system. The vehicle control system is presented in Section 4. Section 5 describes the V2V communication module. Sections 6–8 briefly describe the high-level control, decision making, and the perception and sensor-fusion module. In Section 9 the preparatory simulations and work are described whereas in Section 10 results from post-competition analysis are presented. Finally, Section 11 concludes the paper and suggests directions for future work.

2 Scenarios

Besides the overall goal of GCDC – to boost the introduction of cooperative and automated driving – the scenarios in GCDC are designed to also demonstrate the current development within cooperative intelligent transport systems (C-ITS). The scenarios are influenced by suggestions from domain experts as well as proposals from the participating teams.

The first scenario, shown in Figure F.1 on the left, is the cooperative platoon merge. It involves two platoons driving on two adjacent lanes on a highway. The two platoons must merge into one due to an upcoming construction site where one lane is closed. A competition zone is defined as a zone where the vehicles' operations are judged.

The second scenario, the cooperative intersection shown in Figure F.1 on the right, considers a common urban traffic situation, an uncontrolled T-intersection. Whereas the first scenario requires that all vehicles are interacting and communicating, this scenario involves a mixture of non-cooperative (not communicating) and cooperative vehicles. The scenario involves three cooperative vehicles, one vehicle that is approaching a busy road with two other vehicles driving in both

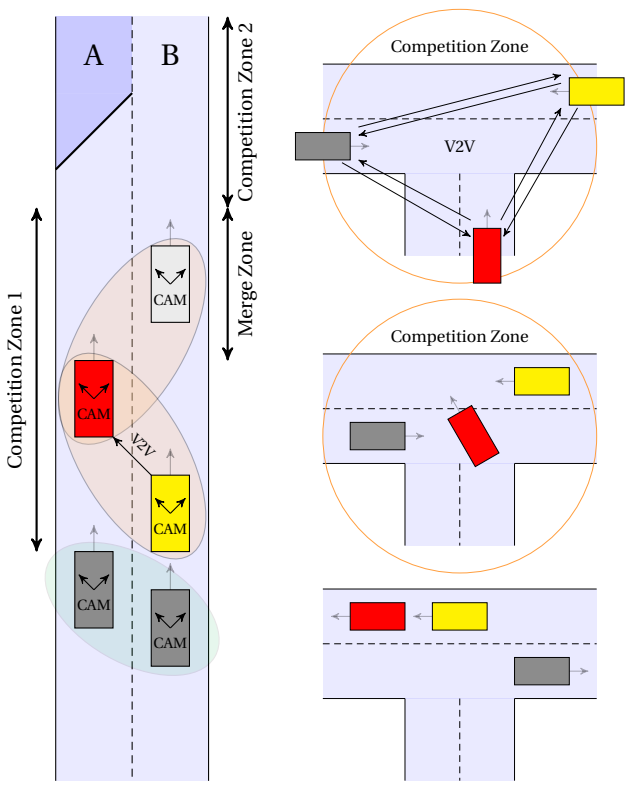


Figure F.1: The schematics of the two judged competition scenarios.

directions. The approaching vehicle transmits its intention, to turn left in the intersection, and the cooperative vehicles on the main road acknowledge its request and help to facilitate the manoeuvre in an efficient manner, i.e., without coming to a full stop. Consequently, the vehicles on the main road help to create proper gaps, allowing a smooth passage for the left-turning vehicle to safely and efficiently cross the intersection. The non-communicating vehicles take only an assumed part in the scenario, virtually following the communicating vehicles on the main road, i.e., only the three mentioned communicating vehicles take part in the challenge scenario.

The third scenario demonstrates an emergency vehicle requiring passage along a highway with congested traffic. This scenario is not part of the judging, yet it is used to demonstrate an everyday traffic situation that needs efficient solutions. Emergency vehicle warning has been considered in the basic set of applications of the C-ITS standard, see e.g. [23] for further details. Since the current version of the emergency vehicle warning only provides a warning about an approaching emergency vehicle it is still confusing for the other road users about where to place their vehicle. With the proposed amendments in GCDC, the emergency vehicle will be able to inform other vehicles of its itinerary and how it wants other vehicles to behave, thereby providing a safe passage.

3 System Architecture

Both on the software and hardware side it was decided to go for a relatively simple solution, for two reasons. First, the general i-GAME concept is to provide robust future solutions for the automotive industry and such a solution fits better with these conceptual demands of the GCDC context. Second, the former Halmstad Team from GCDC 2011 [19] achieved a very good result with a similar setup, thus it was decided to work based on their good experiences.

The base vehicle was a production Volvo S60. The additional control, sensor, communication, and supporting devices mounted in the vehicle (mostly in the trunk) were the dSpace MicroAutoBox (MAB) real-time controller[5], Trimble differential GPS, a radio computer (see Section 5), a Nexus 9 tablet, and a router. The only gateway to the S60 systems was through the dSpace MAB that intercepts car's CAN bus messages and is able to inject additional ones. No other devices were used, in particular the only sensors used in our system were the the radar system of the vehicle monitored through MAB and the Trimble GPS. Finally, the high-level control and coordination of the system was done with a regular laptop, see below.

3.1 Power and CAN bus Arrangement

The power supply arrangement is shown in Figure F.2. The main power source was the car battery of the S60. To operate the 220V equipment a DC to AC inverter was used and to avoid power failures an UPS (Uninterrupted Power Supply) was installed. The 12V equipment such as the CAN bus interface and mode signalling roof lights were powered from the battery. To disconnect the power supply in an unexpected

[5] `https://www.dspace.com/en/pub/home/products/hw/micautob.cfm`.

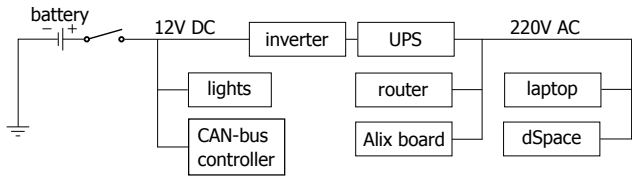


Figure F.2: Power supply setup for the system.

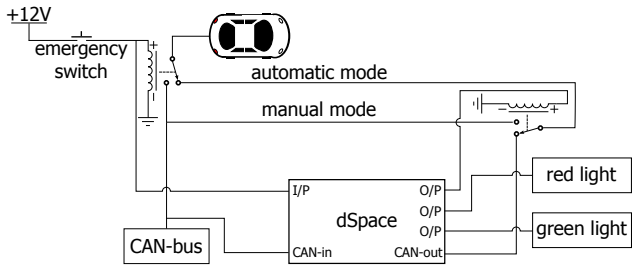


Figure F.3: CAN bus interface diagram.

situation the main power bus was controlled by a driver-side switch. The Trimble GPS has its own battery, thus no external power was needed other than periodically charging the battery.

A custom made CAN bus interface depicted in Figure F.3 was used to switch between automatic and manual driving mode. The emergency button had the highest priority on the bus. Automatic to manual mode transition was also controllable from the MAB, but with lower priority than the button.

Otherwise, all the devices comprising the system were inter-connected through an Ethernet router with cables, with the exception of the Human-Machine Interface tablet that was connected to the car network through wireless communication.

3.2 Software Modules

The software architecture of the system is characterised by its modularity. The system is split into modules as shown in Figure F.4. Each module can be executed independently and on different hardware units. They communicate using Lightweight Communications and Marshalling (LCM) [24], which provides a programming language agnostic solution to communication that abstracts from the actual media, in our case UDP packets in the local network.

The communication (COM), data source (DS), high-level control (HLC), and mid-level control (MLC) modules are implemented as Java applications running on a regular, non-real time Java Virtual Machine:

- COM implements the ITS-G5 communication stack and services used to send and receive V2V messages;
- DS performs sensor fusion with the information received from the COM module

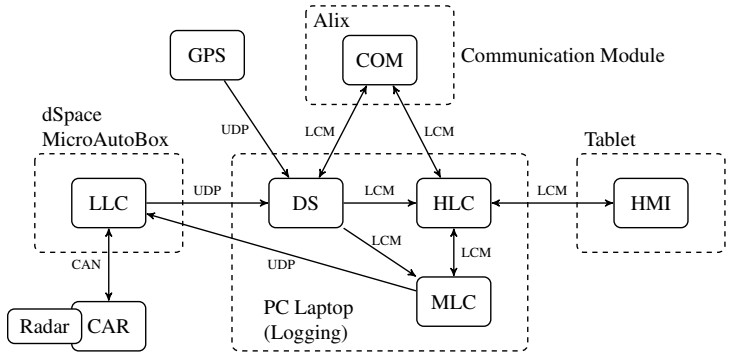


Figure F.4: Software system architecture.

and LLC module (see below) directly from the car and the GPS, and provides this information to the other modules;

- HLC makes high-level decisions regarding manoeuvres following the competition interaction protocols;
- MLC calculates parameters for the LLC speed controller.
- LLC is the speed controller, described in the next section.

The low-level control (LLC) is a Simulink model executed on the dSPACE MicroAutoBox connected to the CAN bus of the car. This controller fully replaces (bypasses) the factory adaptive cruise control system. Finally, the human-machine interface (HMI) Java application runs on an Android tablet. It collects the driver's input regarding configuration for the scenario and asks for confirmation before performing manoeuvres autonomously. It also provides information regarding the status of the scenario and the neighbouring vehicles.

4 Cooperative Adaptive Cruise Control

4.1 Controller Design

The proposed control system strives to utilise the full potential of cooperative driving. It is inspired by the previous Halmstad team solution [19] and [25]. To achieve robustness and modularity, the controller is divided into two layers, mid-level control (MLC) and low-level control (LLC). The MLC communicates with all other processes, determines maximum speed, desired time headway and gathers preceding vehicle information from the sensor fusion module (DS). The LLC keeps the desired distance from the preceding vehicle, and maintains all the constraints provided by the MLC. The LLC strategy is explained below, and its overview is shown in Fig F.5.

The constant time gap (CTG) policy [9] is chosen as the gap regulation policy. According to [9], the time gap, referred to as a *time headway* in this paper, is defined

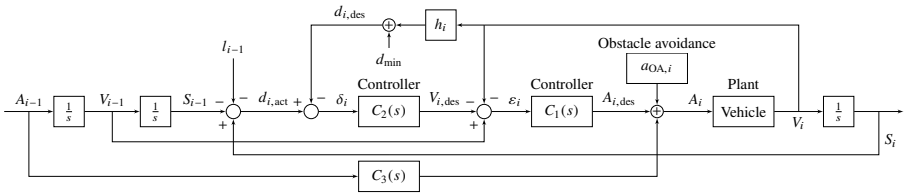


Figure F.5: Overview of the low-level control (LLC) system. $C_1(s)$ is a speed controller, it keeps the speed of the previous vehicle and indirectly regulates the distance. $C_2(s)$ regulates the distance to the vehicle in-front. $C_3(s)$ manipulates the feed-forwarded acceleration A_{i-1}, from the preceding vehicle.

as *"the time between when the rear bumper of the leading vehicle and the front bumper of the following vehicle pass a fixed location on the roadway (measured in seconds)"*. Therefore, the desired inter-vehicular distance for the i^{th} vehicle in a platoon is proportional to its speed, plus a fixed offset (standstill) distance. The desired distance is calculated by eq. (F.1):

$$d_{i,\text{des}}(t) = d_{\min} + h_i \cdot v_i(t) \tag{F.1}$$

where $d_{i,\text{des}}(t)$ is the desired distance (m), $d_{\min}$ is the standstill distance (m), h_i is the time headway (s), and $v_i(t)$ is the vehicle speed (m/s). During GCDC h_i was specified to be 1 s and $d_{\min}$ to be 6 m.

The proposed system is composed of three main controllers:

- C_1, a proportional controller with a lead compensator that acts on speed error ε_i,
- C_2, a proportional-integral controller that acts on distance error δ_i,
- C_3 applies a gain on acceleration from the preceding car as reported through CAM messages.

Since C_1 considers output from C_2, let us first discuss C_2. The distance error δ_i is defined as:

$$\delta_i(t) = d_{i,\text{act}}(t) - d_{i,\text{des}}(t) \tag{F.2}$$

$$\delta_i(t) = S_i(t) - S_{i-1}(t) - l_{i-1} - (d_{\min} + h_i \cdot v_i(t)) \tag{F.3}$$

where S_i represents position of the i^{th} vehicle, and l_{i-1} is the length of the preceding vehicle. The control law for C_2 is:

$$v_{i,\text{des}}(t) = K_{P2} \cdot \delta_i(t) + K_{I2} \int_0^t \delta_i(t)\, dt \tag{F.4}$$

where $v_{i,\text{des}}$ is the desired speed, $K_{P2} = 2.9497$, and $K_{I2} = 4.3615$ (the gain parameters where chosen experimentally, see below).

The controller C_1 then acts on the speed error, which is:

$$\varepsilon_i(t) = v_{i,\text{act}}(t) - v_{i,\text{des}}(t) = v_{i-1}(t) - v_i(t) - v_{i,\text{des}}(t) \tag{F.5}$$

where $v_{i,\text{des}}$ is calculated in eq. (F.4). C_1 is a proportional controller with a lead compensator given by:

$$a_{i,\text{des}} = K_{P1}\left(\varepsilon_i(t) - 7.5e^{-10t}\right) \quad \text{where } K_{P1} = 0.872 \tag{F.6}$$

As part of a safety feature, the Obstacle Avoidance (OA) controller with a potential function according to eq. (F.7) is used to increase deceleration, in case of the preceding vehicle instantaneously applies high deceleration:

$$a_{\text{OA},i} = \begin{cases} -\beta(\alpha d_i + 1)e^{-\alpha d_i} & a_{i-1} < 0 \text{ and } d_i < d_{i,\text{des}} \\ 0 & \text{otherwise} \end{cases} \tag{F.7}$$

where β is a gain factor which indicates the maximum effort of the controller when the distance goes to zero, α is a fall-off rate when the preceding vehicle is getting away, and d_i is the distance to the preceding vehicle. This equation is similar to the OA controller stipulated by the competition organisers [26]. However, apart from activating when the preceding vehicle is decelerating (as in [26]), an added condition is when the actual inter vehicular distance is shorter than the desired distance. Therefore, the OA is applied to facilitate the braking only when these two conditions are true. Because of this, we also decided to amplify the effort of the OA once it engages, the particular parameters we used were $\alpha = 0.3$ and $\beta = 30$, while the organisers proposed $\beta = 3$. Figure F.6 shows the comparison of characteristics of the OA function with our β control parameter and the suggested one. Therefore, the complete acceleration input to the plant is formulated as:

$$a_i = a_{i,\text{des}} + a_{\text{OA}}, i + K_{P3}a_{i-1} \qquad -2 \le a_i \le 2 \tag{F.8}$$

where $K_{P3} = 0.4981$, and the final value a_i is bounded by the maximum acceleration and deceleration, which is -2 to $2\,\text{m/s}^2$ according to the GCDC rules [27].

4.2 Controller Evaluation

During the development phase of the CACC controller the performance was evaluated using Matlab simulations according to the following criteria:

- Performance: to what extent the system is capable of keeping the desired distance to the preceding vehicle and does the system maintain the string stability condition;
- Safety: the system is considered as safe if the actual distance is larger or equal to the desired distance:

$$\begin{array}{ll} d_{i,\text{act}} \ge d_{i,\text{des}} & \text{safe,} \\ d_{i,\text{act}} < d_{i,\text{des}} & \text{unsafe,} \\ d_{i,\text{act}} < d_{\min} & \text{risk of collision.} \end{array}$$

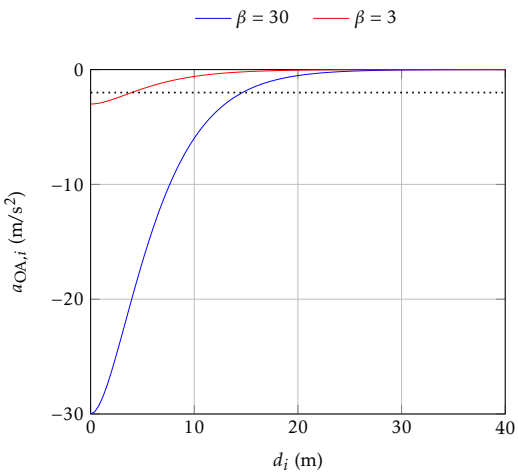


Figure F.6: Plot for OA with different β with black dotted line showing maximum (cut-off) deceleration of $-2\,\mathrm{m/s^2}$.

- Comfort: the smoothness of the controller is measured by the jerk effect, which ideally should be zero: $\ddot{a}(t) = 0$.

It was only after the competition that we were able to evaluate the controller in a realistic setting with the actual competition heats data. This is further discussed in Section 10.

In theory, according to the final value theorem, using just the proportional controllers would be sufficient, i.e., the errors are eventually brought down to zero in steady state conditions. Actual experiments exhibited distance lagging and prompted the introduction of the integral component in C_2 to decrease the reaction time on δ_i. A derivative component would dampen the behaviour and provide smoothness, however, in the rather steady state conditions of GCDC it was not necessary and the tedious tuning of the derivative gain was avoided.

The controller gain parameters $K_{\{P1,P2,I2,P3\}}$ were tuned first approximately with the SISOTOOL from Matlab, and then by experimentation with the organiser team during the competition preparation week. The distinguished feature of the proposed control strategy is the feed forwarding of the acceleration of the preceding vehicle A_{i-1} obtained from the MLC module via CAM messages. This is the point where the cooperative character of the controller is exhibited. The acceleration is manipulated with the gain in C_3, and feed-forwarded to the vehicle. The controller also has the option to use *intended* acceleration of the preceding vehicle (if available) rather than the actual. During brief GCDC off-line experiments the use of the intended acceleration was successfully added to the controller. However, during competition heats the intended acceleration of participants was often either unavailable or faulty, hence the actual acceleration was used during the competition to achieve robustness. Our experimentation result from using the intended acceleration as well as a brief evaluation of the controller are further discussed in Section 10.

5 Communication and Logging Modules

5.1 Communication

The communication system for V2X is composed of: (a) a radio module to handle the physical and data link layer of the communication stack; and (b) a computer to handle the rest of the layers, from network to application layer.

The hardware used for the radio module is a Wistron DCMA82 with an Atheros AR922X chipset attched to an ALIX 2D13 system board, containing 256 MB of RAM and a AMD Geode LX800 processor at 500 MHz. The decision to use this particular board was purely pragmatic – during a communication workshop organised in Sweden two other teams reported it to be capable of meeting the competition requirements. These two teams (from Chalmers) were also geographically nearest which enabled mutual pre-competition testing and support. The computer module used for the upper layers features 8 Gb of RAM and an Intel Core i5-5300U at 2.3 GHz. Besides running the communication module, this computer also executes most of the other components in the system, as illustrated in Figure F.4, and is connected to the rest of the system via Ethernet, using the User Datagram Protocol (UDP) to Ethernet conversion daemon (*udp2eth*)[6].

According to the Open Systems Interconnection (OSI) model, the physical and data link layers are implemented as a modification of the *ath9k* Linux kernel driver, which can be found on GitHub[7]. The modified drivers were loaded in Voyage Linux[8], a lightweight Debian-based Linux distribution. The remaining layers (network, transport, session, presentation, and application) are encapsulated in a Java application. The system uses the GeoNetworking and ASN.1 UPER encoder/decoder implementation by Alexey Voronov [28].

The communication module works independently and transparently to the rest of the system. It decodes and relays information to and from the other modules, and transmits information as V2V messages of the following three types:

- Cooperative Awareness Message (CAM), containing vehicle status information such as position, movement, and other sensor data [29];
- i-GAME Cooperative Lane Change Message (iCLCM), containing information required to perform manoeuvres during the competition [30]. Similarly to CAM, iCLCM-s are sent periodically and contain scenario control flags for the execution of the competition (e.g., start of scenario), platooning information (platoon identifier, desired acceleration), merge scenario information (merge requests and confirmations, pairing arrangements), and intersection scenario information (vehicle identifier and intention);
- Decentralised Environmental Notification Message (DENM), used to notify other users of events such as dangerous road conditions and emergency situations [31].

[6] `https://github.com/jandejongh/udp2eth`.
[7] `https://github.com/CTU-IIG/802.11p-linux`.
[8] `http://linux.voyage.hk`.

The communication module interacts with the system via LCM messages. Two LCM channels (input and output) are used for each type of V2V message. Whenever a module wants to transmit a message, it sends the information on the output LCM channel for that specific message type, CAM, DENM, or iCLCM. Similarly, if a module wants to obtain the information from a given type of message, it can listen to the corresponding LCM input channel. The data types used in these LCM channels contain the same fields and use the same units, as defined in the European Telecommunications Standards Institute (ETSI) standards for CAM and DENM, and as defined by the i-GAME project for iCLCM.

A container class with the latest information required to construct CAM and iCLCM messages is stored in the communication module and updated when new information is received via LCM. For GCDC 2016, the organisers defined the update frequency of CAM and iCLCM to be 25 Hz. Therefore, every 40 ms this information is used to construct the messages, encode them and send them via a Basic Transport Protocol (BTP) socket. Although CAMs and iCLCMs are sent with the same frequency, they are generated with an offset of half a period, i.e. 20 ms, in order to spread the computation load over time and avoid peaks when the messages are generated.

A separate thread receives the messages. Packets are extracted from the BTP socket, decoded, and equivalent LCM messages are created that are broadcast inside the system by placing them in the corresponding LCM channel, through which listening modules receive their own message copies.

No particular signal quality was required by the organisers (e.g., in terms of signal to noise ratio), apart from the frequencies mentioned above and the bi-directional communication distance of 200 m when no obstacles are present [32]. During preparatory tests with the other teams the communication range was verified to be at just this distance in plain sight using small antennas. At the competition site, even using large antennas, the communication was occasionally disrupted at one particular spot during the merge scenario due to a bridge and by the presence of tall vehicles in the platoon. However, at that point no ad-hoc solution could be provided by any of the teams to improve the communication quality, apart from vehicle control fallback procedures. As for the throughput, during the competition it was necessary to process communication from only 10 participating vehicles, hence a simple message dispatching system based on FIFO queues was sufficient to manage the communication with the required frequency. In a more realistic scenario FIFO processing would not be sufficient and possibly cause message buffering congestion, thus after the competition we developed a message prioritising and filtering system [21], outlined in Section 10.

In general, the ITS-G5 standard itself does not require packet delivery guarantees, it is the responsibility of the higher level application (in this case the GCDC iCLCM interaction protocol [30, 33]) to provide safe communication. In particular, low-level packets are not acknowledged, thus it is not possible to state how many of our packets were received by the participants at all times. However, other competition safety requirements were related to the accuracy and delay of the transmitted positioning and velocity data, see Section 7.

5.2 Logging

The logging facility of the system works besides the communication module and is divided into three parts. The first part, implemented as a Java application, keeps track of selected fields from the CAM, DENM, and iCLCM messages. When a message is received, the fields of interest (ones required by the competition requirements, e.g., current position, speed, etc.) are written into a Comma Separated Values (CSV) file. Each line corresponds to one message and it is timestamped with the ETSI Timestamp [34]. The received and transmitted data were stored in separate files and conveyed to the GCDC organisers for judging purposes.

The second part of the logging is done by Wireshark, a network protocol analyser [35] capturing all network traffic and saving it in a packet capture (pcap) file. These files were used after the competition to analyse the performance of the communication protocol, see Section 10.

The last part of the logging system records the LCM traffic, all input and output LCM messages between the different modules of the system are logged. These logs can be replayed to analyse the behaviour of the system off-line.

6 High-Level System Control

The HLC's decision making is implemented as a finite state machine (FSM) that directly follows the GCDC interaction protocol specification predefined by the organisers in [33] and [12]. The events for triggering the state transitions of the FSM are depending on the interaction with the other vehicles, the confirmations via the HMI, and the internal state of the vehicle control system. Confirmations of the driver have been included for safety reasons and to increase the driver's situation awareness in an automated vehicle. An example of a state transition shown in Figure F.7 is when the system is in the S1_PA_WAIT_START state – the ready vehicle is waiting at standstill – and the message with a startScenario is received from the RSU triggering the transition start_a to launch the vehicle. For safety reasons (the vehicle is about to move by itself), the system goes into an intermediate confirmation state S1_PA_CONF_START that sends a request to the HMI and waits for the driver's response before going into the actual vehicle platooning state S1_PA_PLATOON_80. The confirmation steps are skipped for less critical transitions, or when the driver causes an unacceptable delay, most notably during the execution of the intersection scenario.

The state transition rules are part of the *World* – a processing context where the information about the surrounding environment and its state is kept. Information about every vehicle sending CAM or iCLCM message is put into the *World*. This data is fused into one object describing the vehicle within the perception module (within Data Source – DS, see Figure F.4) that was developed by the team for the competition to support the HLC. Furthermore, the concept of a Trust System (TS) was developed to support decision making in an *untrusted* environment. We describe both in the following sections.

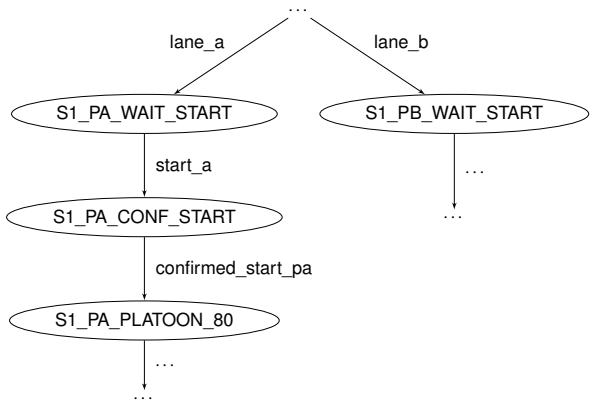


Figure F.7: Sample state transitions from [33].

7 Perception and Sensor Fusion

The high-level controller needs the knowledge about the surrounding environment. The vehicle perceives this environment with the built-in front radar detecting a single target in front of the vehicle, the RTK-GPS device, and the information provided via V2V communication. To build the knowledge about the environment two models are used – vehicle distance model (VDM) and vehicle position model (VPM).

7.1 Vehicle Distance Model

The VDM was designed to describe the relation between the measured radar distance to the car in front and the calculated distance by using the geographical position of the ego (from GPS) and the preceding vehicle (from V2V). The model applies a Kalman Filter, which is proposed in [36], to the distance to the preceding vehicle in case that both distances match each other given a certain boundary (1 m). When the difference between these two ranges exceeds the predefined threshold, the distance measured by the radar takes precedence and it is broadcast to the other modules of the control system. This approach ensures, for safety reasons, that the vehicle's controller uses the radar information about the closest physical obstacle in front of the ego vehicle when wrong information is received via V2V message exchange.

The VDM has been evaluated against the competition requirements by relative comparison with the organisers' equipment in their reference vehicles. That is, by following each other in a platoon, the reported and measured data was verified by the organisers to stay within pre-defined tolerances [32, 37]: 1 m for the position, 0.5 m/s for velocity, $0.2\,\mathrm{m/s^2}$ for acceleration and deceleration, and 200 ms for communication latency (time from data readout to reception by another vehicle). The accuracy of our own sensors were 1 cm for the RTK-GPS when stationary, and 0.25 m for the radar (for the GCDC applicable short range distance of up to 60 m). Our VDM calculations have satisfied these bounds, however, no precise error measurements were reported to us to evaluate how well we stayed within these bounds. Lacking

a ground canonical reference for vehicle positions, this was the only applicable verification method.

7.2 Vehicle Position Model

The VPM that applies an Extended Kalman Filter (EKF) to the vehicle's position and inertial sensor information to improve its geographical position was designed according to [38]. This position model is also applied to the other vehicles based on their transmitted information. An EKF is split into two updates, the time update (prediction) and the measurement update (correction). At first a state vector, which consists of the east and north coordinates, heading to the north, velocity, yaw rate, and acceleration of the vehicle, has been defined. A motion model based on a simple bicycle model [39] is used to predict the new state during the time update. The computation of the Kalman gain and the update of the state using new measurements is performed during the correction phase. The measurement noise covariance matrix describing the error of the measurements is set depending on the source of the information, e.g., inertial sensors and GPS position or V2V information. Due to computational reasons, VPM is applied only to vehicles identified as important, such as the preceding vehicle. Moreover, the Kalman gain of the VPM has been used as an indicator for sensor accuracy [38, 39].

The surrounding vehicles are identified according to their relative position, e.g., *front-left*, *in front of*, or *behind* the ego vehicle, and put into a map categorised by these relative positions. An illustration of the map is depicted in Figure F.8. This map is broadcast to all other control modules. As the perception of the competition car is limited, in particular in tighter curves, the map is generated by applying two different neighbour identification methods that evaluate the received CAM and iCLCM messages. The results of both methods are combined in order to provide a robust identification of the surrounding vehicles.

The first method sorts the vehicles according to their driving direction and platoon identifier from iCLCM. The relative position category is classified based on predefined angles and the distance to the ego vehicle. The second method discards the platoon identifier and relies only on the relative angles and the distance. The range of the angles for each category is calculated dynamically for each vehicle taking the distance to the vehicle and its dimension into account.

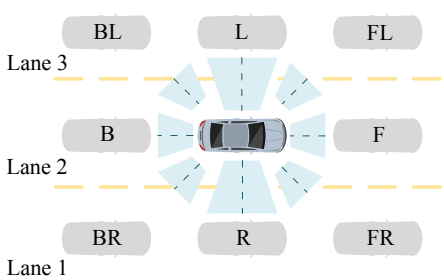


Figure F.8: Illustration of the relative positions in the map.

Each method puts the vehicle with the shortest distance in each category in a separate map. When combining the two maps, the first one where the platoon identifier was used is prioritised, since it is more robust in curves. When there is no vehicle match for the given category, e.g. *front-right*, in the first map, the vehicle possibly identified using the second method is assigned to this category in the combined map. Combining the results this way provides a robust identification of vehicles even if the platoon identifier is missing. Admittedly, it is possible for this procedure to identify one vehicle in two different categories, each resulting from the corresponding map. For example, the first map identifies the vehicle as the *in-front* vehicle and the second map identifies it as the *in the front-left* vehicle. This possibility only occurs when the first map has no vehicle assigned to the given category, *in the front-left* in this case. However, this has no negative impact on the system and it is safe – it is a clear case of a false positive when an existing vehicle is reported in an additional position, while a false negative of not reporting an existing vehicle in any of the positions would be far more dangerous.

The necessity to develop this two-stage identification system became apparent during testing at the Film and Test Location (FTL GmbH)[9] in Aachen, Germany, and at the Dutch Road Transport Authority (RDW) test track in Lelystad. Both locations have considerably tighter curves than the competition zone and hence exhibiting frequent classification errors when using just one method. After introducing the two-stage method and after these tests we had no more opportunities to verify it in curvy road conditions, only at the competition zone in Helmond, which was comparatively straight providing ideal conditions and at which, by visual inspection, the method provided practical 100% robustness. However, more experimentation would be required to further evaluate the method and the choice of control parameters in other conditions.

8 Trust System

The concept of the Trust System (TS) is to evaluate the current situation based on the ego vehicle's and the other vehicle's trust as well as the trust in the environment. This information is represented as one scalar value, the Trust Index (TI), and can be used by the decision making algorithm to make more robust decisions. For instance, the TI can be used by the decision making algorithm to decide on the distance to the vehicle in front when driving in a platoon.

The TS generates and combines partial TI-s for the sensor quality of the ego vehicle, the sensor quality and behaviour of the other vehicles, especially the preceding vehicle and forward partner, and the environment. The quality of a sensor reflects the sensor's precision and reliability. The TS applies the position model (see Section 7) to important vehicles and additionally applies the distance model to the preceding vehicle, for the reason that the location of this vehicle can be verified with one of the ego vehicles own sensors (the radar).

Further details about the perception module and the TS, including the necessary formulas and parameters, can be found in [22] and [40]. The concept of the TS as well

[9] `http://ftl-germany.com`.

as a prototype have been used during the GCDC. The final TS has been evaluated with the communication data gathered during the GCDC highway scenario heats. The sensor fusion as well as the TS are executed independently within the DS module (see Figure F.4). For reasons to be stated in Section 10.3 the TI-s have not been considered by the HLC during the competition.

9 Preparations and Competition Performance

Despite starting the project early enough (the team was fully formed by the end of September 2015) and acquiring the competition vehicle (December 2015), large efforts were put in during several months to enable and stabilise the CAN interface between the dSPACE MicroAutoBox and the car. In effect, the autonomous control of the car was only available six weeks before the competition, yet with continuing stability issues. Furthermore, only two weeks before the competition the interface to the radar module in the car was functioning. Because of these factors, no attempt has been made to support automatic lateral control of the vehicle, despite the fact that the car does offer limited control of the steering from the factory lane-assist system. Yet, the modularisation of the system allowed us to work on the other parts of the system in parallel to the car interfacing work. In particular, extensive use of PreScan simulation software[10] [41] enabled testing the system without the car or its CAN bus interface.

9.1 PreScan Simulations

During development, the complete system was tested by running simulations. PreScan was used to generate simulations that could provide realistic car information to the system, such as GPS position and radar information. UDP sockets were used to communicate between the simulation environment and the running system. Simulations were useful to test individual modules that use raw information originating outside of the system, e.g., the communication packets. Similar simulations were also used to test the interaction among several modules. For example, the use of the CACC with the processed information provided by the communication module.

Vehicle-in-the-loop tests were executed by using data provided by simulated vehicles. This method was particularly useful to test and evaluate the CACC controller. Using a simulated vehicle as the preceding vehicle for the CACC evaluation, eliminated the risk of accident if a real car had been used and a serious error in the controller had occurred.

Lastly, complex simulations involving several vehicles were executed to test the interaction protocols for the highway and intersection scenarios. Multiple instances of the system were executed simultaneously, each one controlling a different simulated vehicle.

[10] `https://www.tassinternational.com/prescan`.

9.2 Competition Time

The system was tested and modified at the Automotive Campus in Helmond during the competition preparatory week of May 20–27 2016. In particular, we developed the two-stage vehicle identification on spot, see Section 7. During the actual competition only minor changes were made. Apart from pre-testing with the Swedish teams at AstaZero[11] in late April 2016, the preparation week was the first time when the interaction and communication could be tested against complete systems from other teams. During that time we could fine tune the CACC controller and fix other encountered issues, however, not all of them, like the communication issues suffered by all teams that were caused by the local surroundings, see Section 5.

10 Post-Competition Evaluation

The main ideas and solutions for the three building blocks of the proposed system have been developed, evaluated and tested using Matlab and PreScan simulations, and by single try-outs with other teams during the pre-competition meetings at the IDIADA testing ground[12] in Spain (4 days) or at the AstaZero test track in Sweden (3 days). During this mutual testing, none of the competing systems were fully developed. It was only at the actual competition week that enough data was collected from sufficiently working systems to evaluate the solutions in a realistic setting. Concretely, the following evaluation aspects were considered:

- practical performance of the proposed CACC implementation in terms of string stability, safety, and comfort,
- robustness of the radio communication implementation against the demands of communicating with 9 other teams without (major) disruptions,
- the Trust Index distribution of TS in a realistic setting.

Apart from collecting the heat logs during the actual competition, the preparatory week was also used to do trial with other teams for experimentation with new features. In particular, with the A-Team from Technical University Eindhoven, the CACC controller was tested based on the intended acceleration as input rather than the actual acceleration.

10.1 CACC Performance

The main concern in evaluating a speed controller is the string stability [42]. As noted in [43], due to the number of control parameters, a proper analysis of string stability for a CACC controller is practically difficult. The particular transfer function of our controller, under simplifying assumptions (no actuation or communication

[11] http://www.astazero.com.
[12] http://www.applusidiada.com.

delays, $d_{\min} = 0$, and $l_{i-1} = 0$) is given by eq. (F.9):

$$\frac{V_i(s)}{V_{i-1}(s)} = \frac{C_3(s) \cdot s^2 + C_1(s) \cdot s + C_1(s)C_2(s)}{s^2 + (1 + C_2(s)h_i)C_1(s) \cdot s + C_1(s)C_2(s)} \tag{F.9}$$

Thus, similarly to [43], our CACC controller has been evaluated using Matlab simulations where an arbitrary number of vehicles with different test parameters could be tested. With these simulations, our controller already proved to perform slightly better than the Sliding Mode Algorithm (SMA) [44]. The simulations were performed against an assumed simple vehicle model which is parametrised by an engine time constant (vehicle's reaction time to acceleration and braking) and sensor (communication) delay, moreover, the obstacle avoidance function was not considered. As an example, Figure F.9 shows a comparison of distance error propagation between our controller (top) and an instance of the SMA controller (bottom) under fixed headway time of 1 s, fixed sensor delay of 0.1 s, and a varying engine time constant of eight vehicles between 0.2 and 0.6 s, i.e., in a simplistically heterogeneous platoon. Both algorithms are string stable in this case (the distance error does not amplify as it propagates backwards to the following vehicle), however, our controller shows a significantly smaller amplitude of the distance error (max. ≈0.2 m) compared to SMA (max. ≈0.6 m) as well as quicker convergence over time. Considering the communicated acceleration feed-forward capability of our controller that SMA naturally lacks, this advantage is rather not surprising, moreover, given a simplistic vehicle model and idealised simulation scenario, hardly conclusive for a realistic setting.

The competition data provided a more realistic evaluation target, however, it was only possible to evaluate the CACC controller against just one vehicle, the currently preceding vehicle (most important object – MIO) in the platoon. The fact that the preceding vehicle is changing during the highway merging scenario from the initially followed vehicle to the merging one essentially introduces additional dimension to the test, as effectively the speed to follow momentarily changes in a non-continuous fashion, while in simulations only an ideal platooning scenario was considered.

Figure F.10 shows the relationship of speeds between three vehicles merging during one of the slow-speed heats, with the vehicle arrangement shown in Figure F.11. The Halmstad vehicle followed the OPC2 (Organiser Pace Vehicle) reference vehicle (id3) and made gap for OPC1, vehicle (id2), to merge in at around timestamp of 540 to 620 s. The drop in speed of the ego vehicle at ≈500 s is obviously intentional to make the gap for id2, after which the ego vehicle immediately follows the new vehicle id2. The corresponding speed error to the preceding vehicle is shown in Figure F.12. Apart from gap-making, the error never exceeds 1 m/s and stays within 0.5 m/s margin most of the time, while it is practically at 0 in stable speed conditions. Moreover, the recorded jerk of the ego vehicle during this heat was within the 0.3 m/s^3. Further details and evaluation of the CACC controller can be found in [45].

Apart from evaluating the base version of the CACC controller with the competition heats, a brief experimentation with the TU/e's A-team during the preparatory week was also performed to preliminarily evaluate the controller that uses the communicated intended acceleration from the MIO rather than the actual acceleration. This was done with the hope that the lag caused by vehicles' dead time could be further reduced. That is, in practice the vehicles should be able to synchronise better

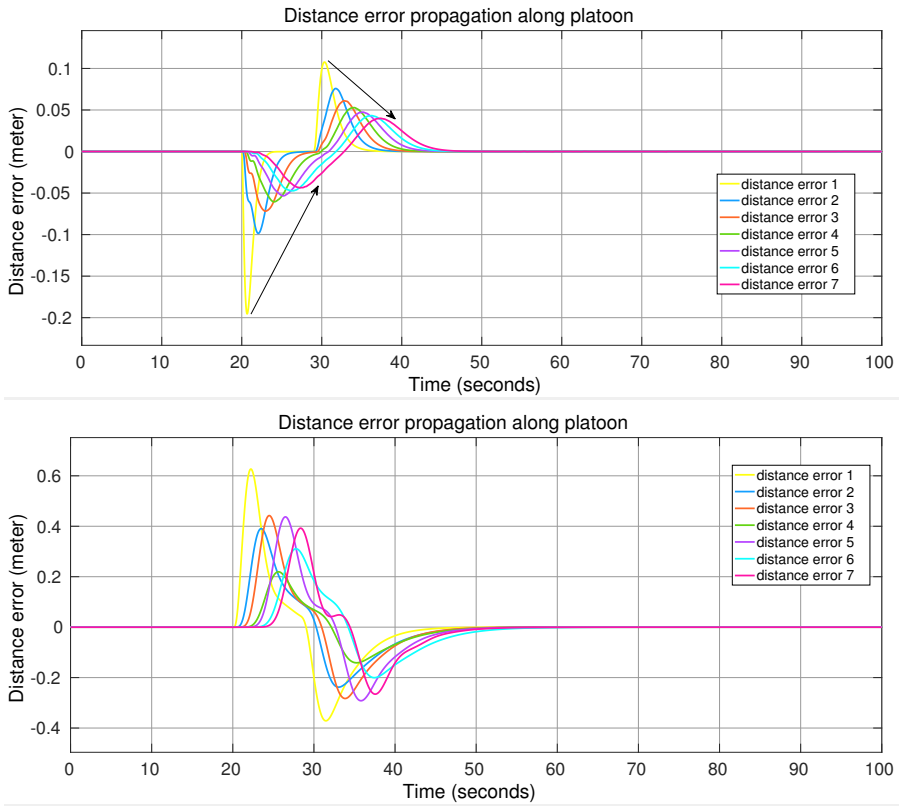


Figure F.9: Distance error propagation comparison to the SMA controller in a heterogeneous platoon simulation.

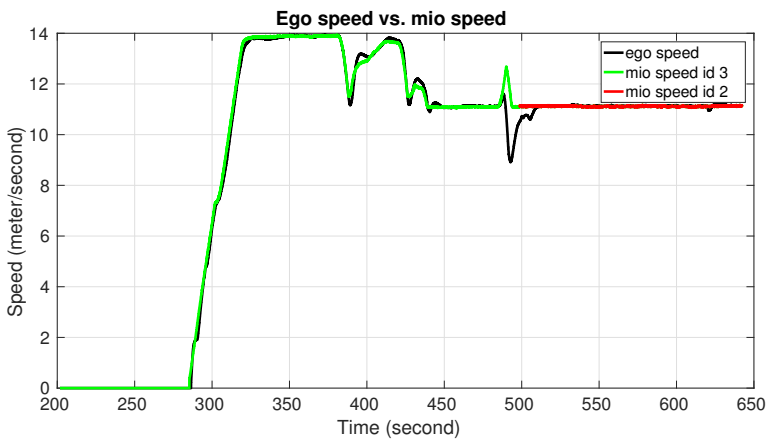


Figure F.10: CACC vehicle speeds during the merging heat.

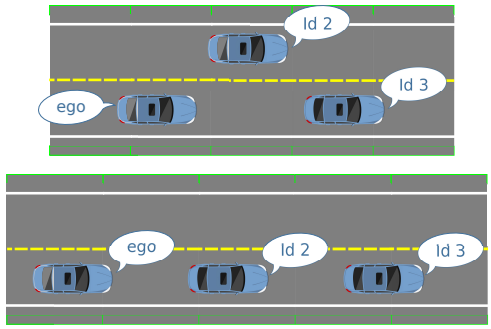


Figure F.11: Vehicle arrangement during the merging heat.

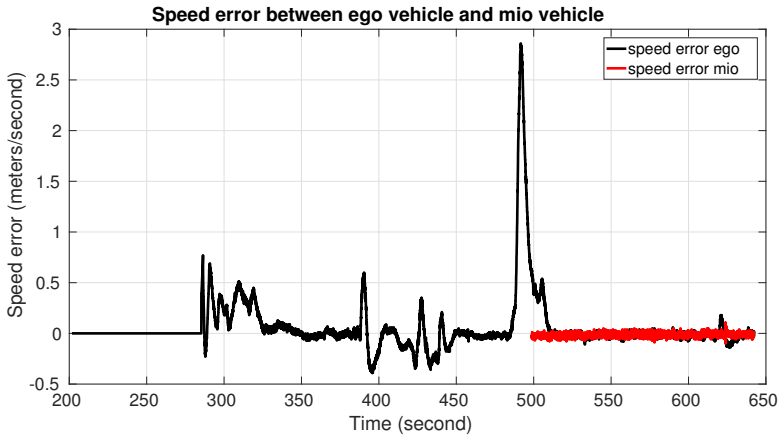


Figure F.12: Speed error for the speeds presented in Figure F.10.

on mutual acceleration. Figure F.13 shows a 1 minute snapshot of the resulting accelerations of the two vehicles and the recorded distance. This single experiment is not sufficient to evaluate this approach (in particular, w.r.t. string stability), nevertheless, the graphs further visualise the behaviour of our CACC controller. During the first phase while going with equal speeds, the ego vehicle matches the desired distance to MIO almost perfectly. When the MIO accelerates (timestamp 168 s), the reaction of the ego vehicle in terms of acceleration is essentially simultaneous and the actual distance drops below the desired one. This is a safe behaviour when accelerating. When the acceleration subsides again (timestamp 183–200 s), the distance error drops. During the braking that follows (timestamp 202 s onwards), the obstacle avoidance function contributes to the deceleration, resulting in the ego acceleration to stay slightly below the MIO acceleration. This allows to increase the gap between the cars and progressively reach the state when the actual distance is above the desired one (from timestamp 208 s on).

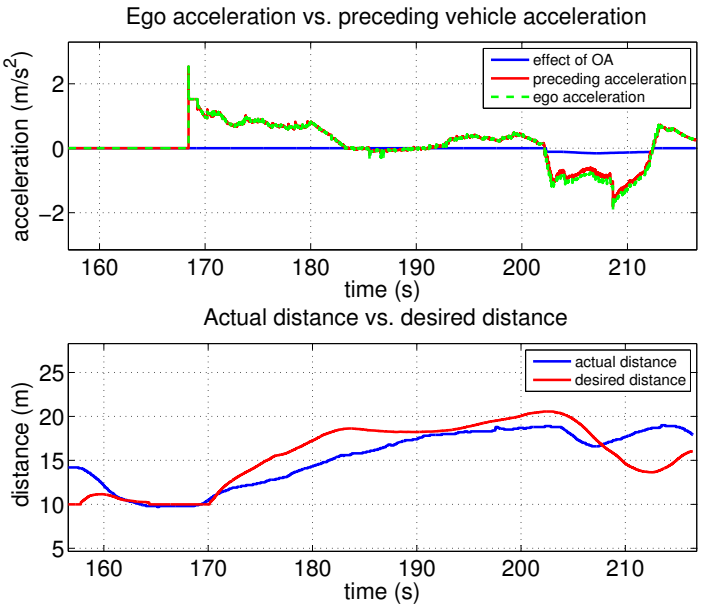


Figure F.13: CACC vehicle accelerations and distance with intended acceleration as input.

10.2 Communication Robustness

The communication module was the first thing developed during the project and it performed very well during the competition and also already during the IDIADA tests in April 2016. In fact, our immediate belief after receiving the final results from the organisers was that the communication robustness, together with complete and requirements compliant logs, were two major contributing factors to our success.

Despite the satisfactory performance of the system during the competition, the communication module has been further developed after GCDC within an MSc project [46]. The main idea is to provide an efficient solution for Vehicular Ad-hoc Networks (VANET) applicable beyond the GCDC setting. The competition scenario involved only 10 vehicles, while a realistic VANET application may involve many more and thus significantly higher communication load. Keeping up with the 25 Hz communication frequency in such a setting becomes a challenge [30].

To this end a packet prioritising and filtering system based on priority queues called Stream-wise Accumulating Priority Queue (SAPQ) has been developed. In short, the streams are identified by the message type (iCLCM, CAM, or DENM) and origin (vehicle identifier), and each message in the stream has a dynamic accumulating factor b_c based on the message type, physical distance of the message origin, taking into account predicted position due to the processing time overhead and the speed of the sending vehicle. Then, the instantaneous priority q_s of the message in the stream s at time t can be given with the following accumulating priority queue formula [47]

$q_s(t) = (t - t_s) \cdot b_c$, where t_s is the arrival time of the first message received in the stream since the last time it was served. Then, in each stream fresher massages override the older messages, which are simply dropped. Out of all streams, the message with the highest priority is served first. This substantially improves message waiting times for the most important messages and prevents message congestion.

To give an idea of the improvement that SAPQ provides, Figure F.14 shows a comparison of the progression of packet waiting times under high system load (300 vehicles) during a simulation. While the simple FIFO implementation constantly increases the waiting time for all of the packets, the SAPQ implementation suffers only from a temporary increase of waiting times during a simulated DENM message outburst, yet with Class 1 packets not exceeding the waiting time of 380 ms. Under low system load (100 vehicles), the waiting times of high priority packets are considerably lower in the SAPQ implementation compared to an average time for unclassified FIFO, see Figure F.15. Finally, Figure F.16 shows the waiting times of the actual communication messages collected during the competition in an otherwise simulated high load setting. Further technical details and analysis of the SAPQ solution can be found in [46] and a companion paper [21].

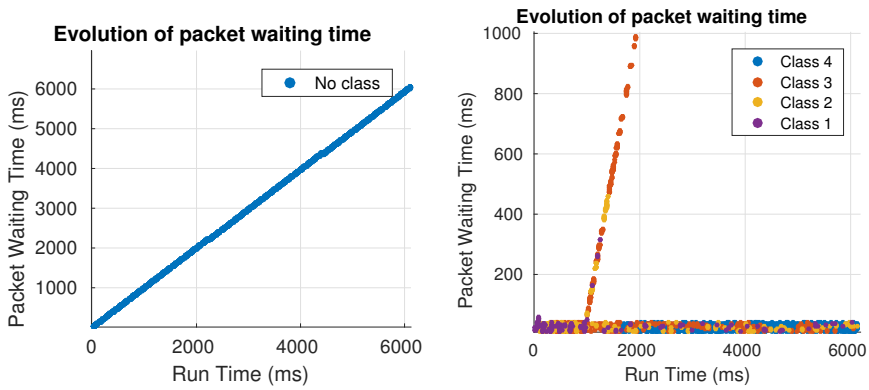


Figure F.14: FIFO (left) vs. SAPQ (right) packet waiting times (high load).

10.3 Trust Index Visualisation

The TS and TI calculation to support decision making described in Sects. 6–8 have been developed for GCDC, but the TIs have not been used to support decision making during the competition for two reasons: (a) due to limited testing possibilities, no prior experimentation data was available to properly weight the decision making based on TIs, the competition data from the fully running systems was the first data available; (b) according to the competition requirements, most decisions had to be driver-confirmed, effectively taking full automation, for which TS would be crucial, out of the scope. In other words, regardless of the calculations a very low trust in the surrounding environment was assumed due to experimental context. Nevertheless, the TS building blocks, in particular the vehicle distance and vehicle position model

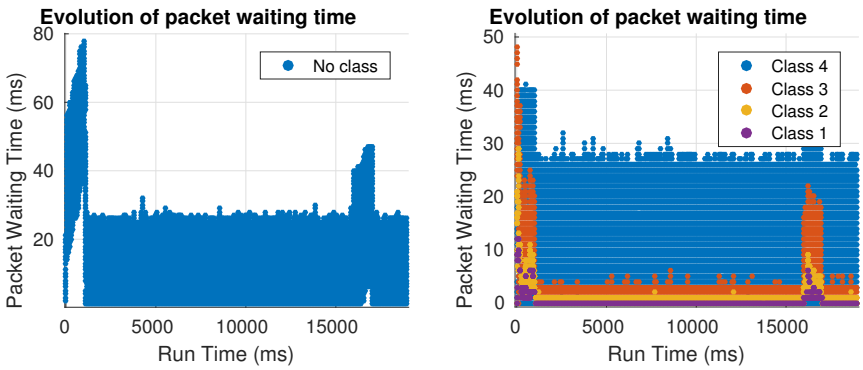


Figure F.15: FIFO (left) vs. SAPQ (right) packet waiting times (low load).

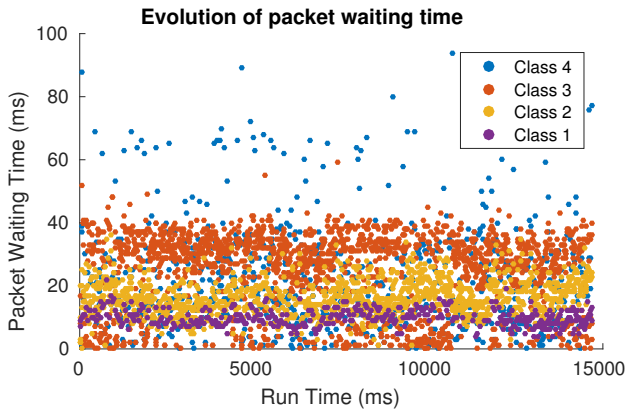


Figure F.16: SAPQ GCDC packet waiting times (high load).

for sensor data fusion, have been implemented and were running live in the system which produced the necessary data for further evaluation of the TS results.

Figure F.17 shows the TI distribution from two highway heats. In the first heat the vehicle was merging from the left platoon, in the second one the vehicle was on the right making a gap. In the first case TI_{MIO} is degrading due to losing the MIO out of sight. After the merge (342 s) the TI_{MIO} re-establishes itself at ≈0.9 level when the system identifies a new MIO. In the second case TI_{MIO} is initially similar to the first case, while it drops after the merge is completed. In this particular case the new MIO did not provide correct position through V2V, only occasionally which is represented by the spikes in the graph. In effect, the global TI is smaller by ≈0.1 compared to the first case indicating overall lower situation awareness. TS information could be, e.g., used to decrease the headway time to the MIO following the high trust in the reliability of its data. Further technical details about the TS can be found in [40] and a companion paper [22].

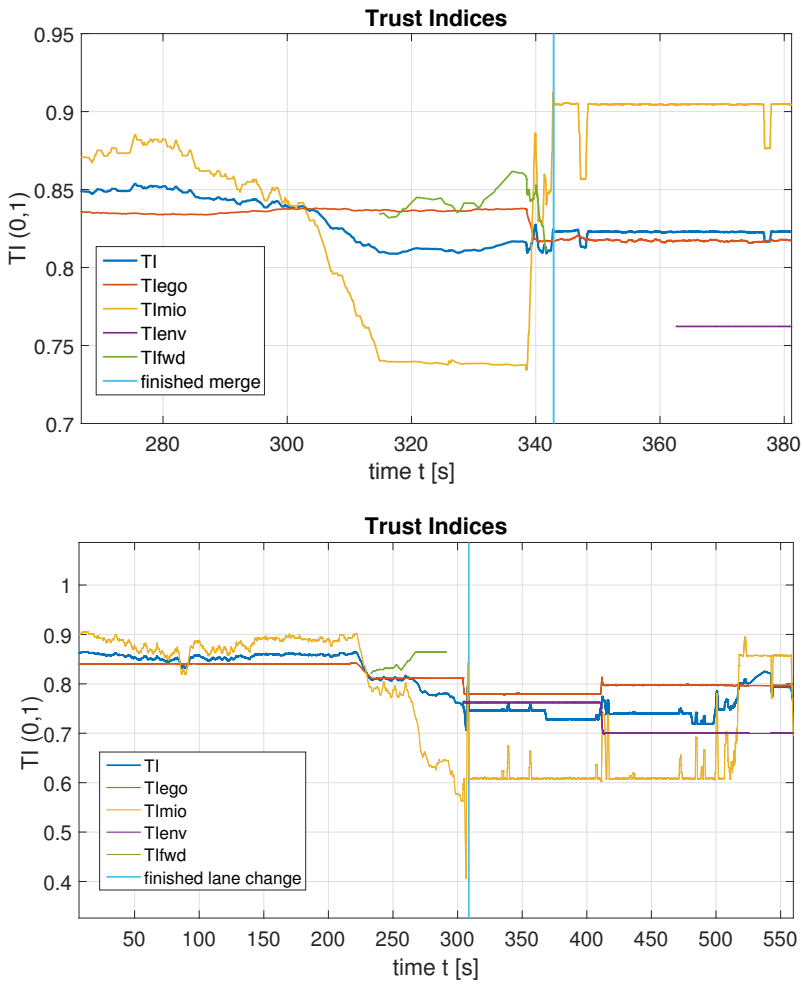


Figure F.17: Trust Index distribution during two merge scenario heats.

11 Conclusions

We have presented the general design and functioning of the team Halmstad system for cooperative driving that was field-tested during GCDC. Apart from successfully competing and winning GCDC 2016, the competition was used as a site for collecting data, which assisted further development, and evaluation of our system during and after the competition. The major contributions and outcomes from the project as a whole are (a) result from the field functioning of the CACC controller and an experiment using intended acceleration for forward feedback in our controller; (b) a priority queue based message dispatching system in the communication module, which substantially improves the communication throughput (further described

in [21]); and (c) the trust system, that was evaluated with realistic data (further described in [22]).

The goal of the i-GAME project that organised the GCDC competition is to address and advance research in intelligent transportation systems keeping the societal challenges in mind. For the Halmstad GCDC student team the main challenge and focus were in the robustness of the system and developing the system in a timely fashion. For these reasons optional functionality was not implemented, in particular the system had no support for lateral control of the vehicle. However, to follow the modern publicity and dissemination trends, two short films advertising the team's efforts were made and published on YouTube[13], one at the preparatory stage of the project, and one after the competition.

An ideal follow-up of the team's effort would be to complete the prototype system with the optional functionalities and continue with the evaluation and development of the single modules to reach production grade quality and applicability in a real traffic environment.

Acknowledgements. We would like to thank the i-GAME project and associated institutions for preparing and organising GCDC 2016, the Swedish CoAct project and institutions for supporting us organisationally and financially, and all of our sponsors including Fengco, TASS International, and Volvo Cars. The work of Wojciech Mostowski is supported by the Swedish Knowledge Foundation grant for the AUTO-CAAS project. We would also like to thank the anonymous reviewers for their valuable comments that helped to improve this paper.

[13] `https://youtu.be/rKvb6Dmy7sY`, `https://youtu.be/2IoVsAl2yTA`.

Bibliography

[1] Eurostat – Statistical Office of the European Communities, *Energy, transport and environment indicators*, ser. Eurostat Pocketbooks on Environment and energy. Eurostat, 2014.

[2] C. Wallmark, M. Goldman, P. Berglund Odhner, S. Forward, and K. Hoyer, "Fossiloberoende fordonsflotta 2030 – Hur realiserar vi målet?" Sweco, Tech. Rep., 2016, in Swedish.

[3] G. C. Patton, C. Coffey, S. M. Sawyer, R. M. Viner, D. M. Haller, K. Bose, T. Vos, J. Ferguson, and C. D. Mathers, "Global patterns of mortality in young people: a systematic analysis of population health data," *The Lancet*, vol. 374, no. 9693, pp. 881–892, Sep. 2016.

[4] W. J. Gillan, "PROMETHEUS and DRIVE: their implications for traffic managers," in *Vehicle Navigation and Information Systems Conference, 1989. Conference Record*, 1989, pp. 237–243.

[5] R. Rajesh, T. Han-Shue, L. Boon Kait, and W.-B. Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 4, pp. 695–708, 2000.

[6] C. Bergenhem, Q. Huang, and A. Benmimoun, "Challenges of platooning on public motorways," in *Proceedings of the 17th ITS World Congress*, 2010, pp. 1–12.

[7] A. De La Fortelle, X. Qian, S. Diemer, J. Grégoire, F. Moutarde, S. Bonnabel, A. Marjovi, A. Martinoli, I. Llatser, A. Festag *et al.*, "Network of automated vehicles: the AutoNet 2030 vision," in *ITS World Congress*, 2014.

[8] M. Aramrattana, T. Larsson, J. Jansson, and C. Englund, "Dimensions of cooperative driving, its and automation," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2015, pp. 144–149.

[9] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Ferlis, "Cooperative adaptive cruise control (CACC) definitions and operating concepts," in *Proceedings of the 94th Annual TRB Meeting. Transportation Research Board*, 2015.

[10] H. H. Bengtsson, L. Chen, A. Voronov, and C. Englund, "Interaction protocol for highway platoon merge," in *IEEE 18th Intl. Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 1971–1976.

[11] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by VANET," *Vehicular Communications*, vol. 2, no. 2, pp. 110–123, 2015.

[12] E. S. Kazerooni and J. Ploeg, "Interaction protocols for cooperative merging and lane reduction scenarios," in *IEEE 18th Intl. Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 1964–1970.

[13] S. Eilers, J. Mårtensson, H. Pettersson, M. Pillado, D. Gallegos, M. Tobar, K. H. Johansson, X. Ma, T. Friedrichs, S. S. Borojeni, and M. Adolfson, "Companion – towards co-operative platoon management of heavy-duty vehicles," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 1267–1273.

[14] A. Böhm, M. Jonsson, and E. Uhlemann, "Performance comparison of a platooning application using the ieee 802.11p mac on the control channel and a centralized mac on a service channel," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2013, pp. 545–552.

[15] M. Segata, B. Bloessl, S. Joerer, C. Sommer, M. Gerla, R. L. Cigno, and F. Dressler, "Towards inter-vehicle communication strategies for platooning support," in *2014 7th International Workshop on Communication Technologies for Vehicles (Nets4Cars-Fall)*, Oct. 2014, pp. 1–6.

[16] European Telecommunications Standards Institute, "Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part," ETSI, TS 102 687 V1.1.1, Jul. 2011.

[17] C. Sommer, S. Joerer, M. Segata, O. K. Tonguz, R. L. Cigno, and F. Dressler, "How shadowing hurts vehicular communications and how dynamic beaconing can help," *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1411–1421, Jul. 2015.

[18] E. van Nunen, M. Kwakkernaat, J. Ploeg, and B. D. Netten, "Cooperative Competition for Future Mobility," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, 2012, pp. 1018–1025.

[19] K. Lidström, K. Sjöberg, U. Holmberg, J. Andersson, F. Bergh, M. Bjäde, and S. Mak, "A Modular CACC System Integration and Design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1050–1061, 2012.

[20] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. Hoang Bengtsson, and J. Didoff, "The Grand Cooperative Driving Challenge (GCDC)

2016 – boosting the introduction of Cooperative Automated Vehicles," *IEEE Wireless Communication Magazine*, vol. 23, pp. 146–152, Aug. 2016.

[21] V. Díez Rodríguez, J. Detournay, A. Vinel, and N. Lyamin, "An approach for receiver-side awareness control in vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1227–1236, 2018.

[22] T. Rosenstatter and C. Englund, "Modelling the level of trust in a cooperative automated vehicle control system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1237–1247, April 2018.

[23] C. Englund, K. Lidström, and J. Nilsson, "On the need for standardized representations of cooperative vehicle behavior," in *Second Intl. Symposium on Future Active Safety Technology toward Zero-traffic-accident.* Society of Automotive Engineers of Japan, Inc., Sep. 2013, pp. 1–6.

[24] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight Communications and Marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 4057–4062.

[25] A. Ebadighajari, "Vehicle control in cooperative driving," Master's thesis, Chalmers University of Technology, Department of Signals and Systems, 2011.

[26] A. M. Medina and E. Semsar-Kazerooni, "DEL150318 i-GAME D2.2 Generic real-time control system FINAL," i-GAME Project, Tech. Rep., Mar. 2015.

[27] C. Englund, J. Didoff, L. Chen, A. Voronov, E. van Nunen, I. Besselink, and A. Morales Medina, "DEL151031 i-GAME D1.4 Final report on requirements specification," i-GAME Project, Tech. Rep., 2015.

[28] A. Voronov, J. De Jongh, D. Heuven, and A. Severinson, "Implementation of ETSI ITS G5 GeoNetworking stack, in Java: CAM-DENM / ASN.1 PER / BTP / GeoNetworking," Jun. 2016. [Online]. Available: https://doi.org/10.5281/zenodo.55650

[29] European Telecommunications Standards Institute, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness basic service," ETSI, TS 302 687-2 V1.3.1, Sep. 2014.

[30] O. Baijer, P. Balaguer, H. H. Bengtsson, L. Chen, L. Garcia-Sol, and J. van de Sluis, "D3.2 Proposal for extended message set for supervised automated driving," i-GAME Project, Tech. Rep., Sep. 2015.

[31] European Telecommunications Standards Institute, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specification of Decentralized Environmental Notification basic service," ETSI, TS 302 637-3 V1.2.2, Nov. 2014.

[32] J. van de Sluis, "D3.1 Wireless communication basic specification document," i-GAME Project, Tech. Rep., Sep. 2015.

[33] E. Semsar-Kazerooni, A. Morales Medina, and H. H. Bengtsson, "DEL150330 i-GAME D2.1 Interaction Protocol," i-GAME Project, Tech. Rep., 2015.

[34] "ISO 8601:2004 Data elements and interchange formats – Information interchange – Representation of dates and times," 2004.

[35] G. Combs *et al.*, "Wireshark – http://www.wireshark.org."

[36] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering*, vol. 83, no. 1, pp. 95–108, 1961.

[37] C. Englund, J. Didoff, L. Chen, A. Voronov, E. van Nunen, I. Besselink, and A. M. Medina, "D1.4 Final report on requirements specification," i-GAME Project, Tech. Rep., Oct. 2015.

[38] N. Magnusson and T. Odenman, "Improving absolute position estimates of an automotive vehicle using GPS in sensor fusion," Master's thesis, Chalmers University of Technology, 2012.

[39] L. Danielsson, "Tracking and radar sensor modelling for automotive safety systems," Ph.D. dissertation, Chalmers University of Technology, 2010.

[40] T. Rosenstatter, "Modelling the level of trust in a cooperative automated vehicle control system," Master's thesis, Halmstad University, School of Information Technology, Sep. 2016.

[41] R. Schubert, N. Mattern, and R. Bours, "Evaluating automated vehicle systems using probabilistic sensor simulations," in *10th ITS European Congress, Helsinki, Finland, 16–19 June 2014*, 2014.

[42] D. Swaroop and J. K. Hedrick, "String stability of interconnected systems," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 349–357, Mar. 1996.

[43] C. Wang and H. Nijmeijer, "String stable heterogeneous vehicle platoon using cooperative adaptive cruise control," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 1977–1982.

[44] J. Zhou and H. Peng, "Range policy of adaptive cruise control vehicles for improved flow stability and string stability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 229–237, Jun. 2005.

[45] O. Uddman Jansson and G. Shahanoor, "Evaluation of string stability during highway platoon merge," Master's thesis, Halmstad University, School of Information Technology, Sep. 2016.

[46] V. Díez Rodríguez and J. Detournay, "An approach for receiver-side awareness control in vehicular ad-hoc networks," Master's thesis, Halmstad University, School of Information Technology, Sep. 2016.

[47] L. Kleinrock, "A delay dependent queue discipline," *Naval Research Logistics Quarterly*, vol. 11, no. 3-4, pp. 329–341, 1964.

Chapter G

Modelling the Level of Trust in a Cooperative Automated Vehicle Control System

Adapted version that appeared in T-ITS 2018

T. Rosenstatter, C. Englund

Abstract. Vehicle-to-vehicle communication is a key technology for achieving increased perception for automated vehicles, where the communication enables virtual sensing by means of sensors in other vehicles. In addition, this technology also allows detection and recognition of objects that are out-of-sight. This paper presents a trust system that allows a cooperative and automated vehicle to make more reliable and safe decisions. The system evaluates the current situation and generates a trust index indicating the level of trust in the environment, the ego vehicle, and the surrounding vehicles. This research goes beyond secure communication and concerns the verification of the received data on a system level. The results show that the proposed method is capable of correctly identifying various traffic situations and how the trust index is used while manoeuvring in a platoon merge scenario.

Modelling the Level of Trust in a Cooperative Automated Vehicle Control System

1 Introduction

Yearly more than 1.2 million fatalities occur on our roads worldwide making traffic accidents to one of the globally leading causes of death [1]. The increasing number of vehicles on public roads and the goal to reduce the environmental impact and improve traffic safety combined with the technological progress leads to research and development in the area of autonomous and cooperative driving. Davila et al. [2] published results from the SARTRE project where they investigated the benefits of platooning systems. Their conclusion was that platooning is safer than manual driving since the vehicle control system and the vehicle dynamics are fully automated. In addition, autonomous vehicles have the potential to increase the comfort for the passengers.

To bring automated vehicles to the market we face considerable safety challenges that may only be overcome by compound research within local awareness, perception and driver support enabled by intelligent vehicular systems [3].

Consequently, software has become the major area of innovation within a vehicle, more than 80 percent of the novelty is achieved by computer systems and their software [4]. The telematics systems of e.g. Daimler and Kia use the Internet for exchanging vehicle status information and calls automatically to the emergency service number in case of an accident [5]. Publicly known projects, such as the self-driving car project[1] from Google, demonstrate the technological progress over the past years.

Automated driving aims to perceive the environment with the on-board vehicle sensors, e. g., Global Navigation Satellite System[2] (GNSS), radar, lidar, and camera systems. Due to the high cost of long range and wide angle sensors and the limitation of the proximity sensors to only being able to detect line-of-sight objects, cooperative driving has turned out to be a reasonable complement. Instead of using expensive high fidelity sensors, cooperative information has the advantage of also perceiving out-of-sight objects via wireless exchange of local information [6]. Consequently, cooperative driving can be the key technology to increase traffic safety and efficiency [7].

One popular application based on Vehicle-to-Vehicle (V2V) communication is platooning, or cooperative adaptive cruise control. For instance, an adaptive cruise control is only following the vehicle in front by measuring the speed of the vehicle and distance to it with the ego (own) vehicle's sensors. It does not significantly

[1] https://www.google.com/selfdrivingcar/

[2] Includes the satellite localization systems GPS, Galileo and GLONASS

increase the efficiency of the entire platoon. With the use of wireless communication to exchange sensor information, e. g., the intended acceleration of the vehicle in front, or the speed, position, and acceleration of the platoon leader, time gaps of less than one second can be achieved [8]. Another field of application is a cooperative interaction to safely cross an intersection [9].

While driving in automated mode the vehicle fully trusts its own sensors. A combination of several sensors in a Sensor Fusion (SF) module [10–12] gives an accurate representation of the local proximity. In a cooperative system where sensor readings from other vehicles are being shared, the vehicles will require methods to incorporate those signals in their own control system. This paper introduces a Trust System (TS) that evaluates the current traffic situation based on the sensor readings from both the on-board vehicle sensors and the surrounding vehicles to support decision making in the cooperative and automated vehicle.

The TS creates a Trust Index (TI) by considering the quality of the sensor information provided by the own vehicle and other traffic participants describing their behaviour as well as the environment itself. The proposed system is inspired by Aramrattana et al. [13] where dimensions of cooperative driving, ITS and automation are described. The proposed TS is designed to handle the surrounding vehicles (number of actors) the environment (individual, local or global scope) and the type of driving task (operational, tactical or strategical). The use of this TS allows the decision-making controller to make reliable and safe decisions when interacting with a specific vehicle or operating in a particular environment. Moreover, a prototype of the TS has been tested and evaluated in a Volvo S60 during the Grand Cooperative Driving Challenge (GCDC)[3] 2016 in Helmond, the Netherlands.

Securing the communication between the vehicles and infrastructure, within the vehicle as well as other security mechanisms are inevitable for future vehicles. This approach has to be considered as a model on the system level that supports the decision making module by indicating the trust in the perceived situation.

The reminder of the paper is organized as follows. Section 2 discusses current research relevant for the proposed model. Section 3 presents the proposed approach. Section 4 describes how the results have been achieved. In Section 5, the results of the experiment using the proposed TS are discussed. Finally, Section 6 concludes the paper and presents directions of future work.

2 Related Work

Recent research within this field has focused on how to establish trust between agents or vehicles within a Vehicular Ad-Hoc Network (VANET). The authentication of nodes is essential for Vehicle-2-Everything (V2X) communication as well as being able to take counter actions to malicious events and actions. The setup of a trustworthy connection between the nodes at the communication level is the base for further security related applications, such as a TS that evaluates the sensor accuracy and the behaviour of the vehicles. The different types of trust establishing techniques are explained in [14]. Zhang explains in [15] techniques to model trust, such as

[3] http://gcdc.net/en/

entity-oriented (e.g. [16] by Minhas et al.), data-oriented (e.g. [17] by Raya et al.), and combined trust models (e.g. VARS in [18]). However, none of these methods deal with the sensor quality, nor the behaviour of the vehicles or nodes. They are focusing on evaluating if nodes communicate the correct events, e. g., slippery road or traffic jam. Models for distributing the trust values between the vehicles are discussed by Agarwal et al. in [19]. Considering the V2V information as a new virtual sensor and estimating a higher accuracy using SF is described in [12]. The result was only verified within a simulation environment. Moreover, Bhargava et al. proposed in [20] a Kalman Filter for calculating a predicted trust. The results show that the computation of the future trust using a Kalman Filter has a problem with time varying trust values. For instance, a constant decrease of the trust value leads to an increasing error of the estimation.

Trust is a common measure in decentralised systems, such as Multi-Agent Systems (MAS). Aras et al. discuss in [21] the relation between trust and uncertainty. The identified sources for uncertainty in MAS are among others, the uncertainty in the observation and the uncertainty when using second-hand information. The authors propose that the trust representation has to reflect the uncertainty and it should allow the use for decision making. One open question is whether it should be represented by a scalar value or be composed of a more complex representation [21].

The contribution of this paper is a model on how to present trust in the own vehicle, the other vehicles, as well as in the environment. Additionally, this model has been tested in an environment with other cooperative vehicles.

3 Proposed Approach

The proposed TS evaluates the current traffic situation by considering different factors, each represented as a partial TI, which are combined into one of the TIs presented in this work, namely the TI about the ego vehicle, surrounding vehicles, and the environment. The generated TIs are broadcasted to all modules of the vehicle's system, which allows, for instance, the decision making module to include this information when making decisions. This section gives an overview of the vehicle system and data acquisition, followed by the identified factors that influence the decision making and the calculation of the TIs.

3.1 System Overview

The results of the TS are evaluated using team Halmstad's GCDC competition car, which provides data from the radar, inertial sensor data, and V2X communication for perceiving the environment. The GeoNetworking protocol defined by ETSI [22] is used for communication. The transmission on the physical layer is established using the standard for wireless vehicular communication, IEEE 802.11p [23] and Cooperative Awareness Messages (CAMs) are used to exchange the sensor information of the vehicle between each other [24].

The on-board sensors of the vehicle establish the base for the situation awareness of the ego (own) vehicle. To calculate the relative position to other vehicles, it is

necessary to use a positioning system via satellites such as GPS (Global Positioning System) or Galileo. Many GPS devices provide additional information about the measured position, the dimensionless Dilution of Precision (DOP) values. Accuracy describes the absolute position error, whereas the location error is expressed by precision. Horizontal DOP (HDOP) and Vertical DOP (VDOP) characterise the precision of the horizontal or vertical position solution. Milbert discusses in [25] the behaviour of these DOP values [26, 27].

Table G.1 lists the variables including their resolution used in the proposed TS. For a more detailed description of the exchanged information see the ETSI and the i-GAME specifications [28, 29]. HDOP and VDOP values are not included in the afore-mentioned specifications and are thus the only measurements in Table G.1 that are not being exchanged. To perform cooperative manoeuvres the ego vehicle needs to interact with the surrounding vehicles. The most relevant vehicle is the preceding one, which will be named Most Important Object (MIO), according to [29].

Providing other vehicles information about oneself enables the design of more efficient manoeuvres, for example a cooperative adaptive cruise controller that reacts smoother to speed changes using the desired acceleration of the MIO [8].

In this work, a vehicle distance model (VDM) is used that describes how the Kalman Filter, presented by R. E. Kalman in [30], is used. The distance to the preceding vehicle can be obtained by two sensors, the vehicle's front radar and the distance between two geographic positions. Consequently, the distance represented by the two

Table G.1: Relevant variables in a VANET environment.

Type	Measurement	Resolution
On-board	speed over ground	0.01 m/s
	longitudinal acceleration	0.1 m/s^2
	desired long. acceleration	0.01 m/s^2
	lateral acceleration	0.1 m/s^2
	yaw rate	0.01 deg/s
Geographical position	latitude	0.1 µdeg
	longitude	0.1 µdeg
	heading	0.1 deg
	HDOP	–
	VDOP	–
With respect to preceding vehicle	bearing	0.002 rad
	range	0.01 m
	range rate	0.01 m/s
	time headway	0.01 s

measurements is linear Gaussian distributed and a Kalman Filter can be used for fusing this data. Delays and latency of the two observations can be considered by adapting the variance according to the delay. The relation between speed, acceleration, yaw rate, heading, and position is non-linear. Thus, it is necessary to create a model that represents the relation between these measurements. Here, a vehicle position model (VPM), based on an extended Kalman Filter (EKF), is used for sensor fusion.

3.2 Vehicle Distance Model (VDM)

The VDM combines the distance to the vehicle in front measured by the radar with the distance using the reported geographical position. This model includes safety mechanisms to provide correct, or at least safe, information to the vehicle's control system. Figure G.1 illustrates the model for fusing the distance to the vehicle in front given two different sources. The geographical position can be the raw data provided by the GPS device or an already filtered signal. The *Distance Calculation* block calculates the ellipsoidal distance using the geographical positions of both vehicles and their length. The position of the vehicle is defined as the geometrical centre of the vehicle.

Sensor data from one of the sensors may be absent due to e.g. environmental changes such as driving in a tunnel or on a curvy road, therefore it is important to choose a proper variance for the data and select the distance to the preceding vehicle with respect to these circumstances. This procedure is performed by the *Data Selection* shown in Figure G.1.

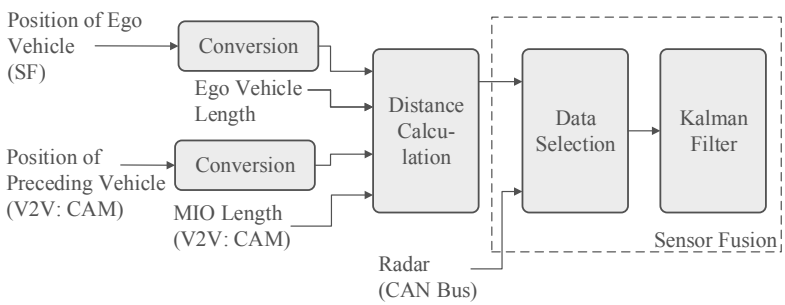


Figure G.1: Vehicle Distance Model – linear sensor fusion.

The *Data Selection* chooses the measurement(s), adapts the variance, and gives this information to the KF module, which acts as a one dimensional filter [31]. Thus, use cases like driving in a tunnel or losing the radar target on a curve are safely considered. Assuming that the distance calculated using the geographical position is above or below a certain threshold compared to the radar information, the system will consider only the radar distance. This procedure ensures that the vehicle does not crash with the car in front when receiving an incorrect position from the MIO.

3.3 Vehicle Position Model (VPM)

The non-linear model, VPM, is applied to both, the ego vehicle's sensor data as well as the sensor data provided via V2V communication by the surrounding vehicles. This model uses the EKF proposed in [32]. However, it has to be highlighted, that the sensor data from the surrounding vehicles cannot be verified with sensors of the ego vehicle.

3.4 Potential Factors Influencing the Decision Making

Automated and cooperative vehicles are introduced to improve traffic safety and efficiency [2], and to reduce greenhouse gases as well as reducing air pollution. However, using other vehicles' information as a basis for decision making in a safety-critical system also brings safety and security concerns with it. In an operational environment, all vehicles might not provide equally reliable information and thus, a system that evaluates the received information is necessary.

The vehicular control system needs to adapt to changing situations as well as the environment. It is therefore desirable that the system can rely on the information provided by the surrounding vehicles, in order to make reliable and safe decisions. The factors that may influence the vehicle's driving behaviour or decision making are described below.

Sensor Quality The knowledge about the precision and accuracy of the surrounding vehicles is important to make profound decisions based on this information. Sensor quality is essential when it comes to the data from other vehicles since the ego vehicle may not be able to verify this information with its own sensors. The CAM contains fields describing the accuracy of the measurements, but it might be the case that the sensor, such as the GPS position, is highly dependent on the environment and thus the message field might not be sufficiently updated.

Static Environment Most of the environment along our roads does not change frequently, e. g., tunnels, bridges, and guard rails are considered as static. The environment has a strong impact on our sensors. As an example, the satellite connection of the GPS device is dependent on the environment, due to the reflection of electromagnetic waves between tall buildings within cities.

Dynamic Environment Loss of messages can be an indicator of an environmental change. Other traffic, for example other temporal objects, such as vehicles can influence the radio communication. During tests with a vehicle able to use V2X communication, it was experienced that trucks can block the communication between the preceding vehicle of the truck and the vehicle in the back, due to the physical properties of a truck. Identifying that there is a temporary communication loss caused by another vehicle is necessary in order to be aware of the current situation.

Behaviour The longitudinal and lateral controller design depends on the manufacturer or even on the model of the vehicle. For that reason, it has to be considered how the vehicle behaves or reacts to certain situations and events. For example, a vehicle following another vehicle that has a high velocity fluctuation should be able to compensate these fluctuations to maintain string stability. One more indicator of

the behaviour is the observation of the vehicles while they are interacting with each other.

The TS takes the aforementioned factors into account and provides the software modules of the ego vehicle with a TI and a map of the surrounding vehicles. The identification of the surrounding vehicles, especially the MIO, is needed to interact with each other.

3.5 Trust Index

The TI is designed for a cooperative automated vehicle control system and is a scalar value that indicates the current overall trust. It combines the trust in the environment, in the vehicle itself, and the trust in the surrounding vehicles. An absolute TI was chosen for the reason that such an index is easier to exchange with other vehicles in case of using a distributed database providing TIs of other vehicles.

To combine different TIs into one integrated index that describes the overall trust in both the environment and the vehicles, a weighted average is proposed. The formula for estimating the weighted average TI is described in Equation G.1.

$$TI = \frac{\sum_{n=0}^{N} w_n \cdot TI_n}{\sum_{n=0}^{N} w_n}, \tag{G.1}$$

where

TI	is the combined TI,
N	is the total number of TIs,
w_n	is the weight for that specific TI, and
TI_n	is the index of a certain TI.

The proposed four TIs are TI_{ego}, TI_{mio}, TI_{env}, and TI_{v_i}. TI_{ego} describes the trust in the ego vehicle according to sensor quality gathered from the VPM, including the knowledge about the DOP values gathered from the GPS device. TI_{mio} represents the trust in the preceding vehicle by combining the sensor quality and the behaviour of the vehicle. The trust in the environment is represented by TI_{env} and the trust in the vehicle i is described by TI_{v_i}. Figure G.2 illustrates the partial TIs and their sources that are combined into one index.

3.5.1 Trust Index TI_{ego}

This TI indicates the trust of the ego vehicle based on its own sensor data. Validating the own trust in the sensor quality is essential in order to create reliable awareness. If the system is not able to trust its own sensor data, the system will not be able to perceive the environment correctly and thus the passenger's safety of the ego and the surrounding vehicles would be at risk. The EKF of the VPM provides the Kalman gain K_k at each time step and reflects how much "trust" it puts into both the predicted state and the new measurement. The DOP values provided by the GPS device indicate the precision of the position.

Figure G.3 illustrates the process of computing TI_{ego}. The first step is to compute a partial TI that contains the quality of the measured position according to the

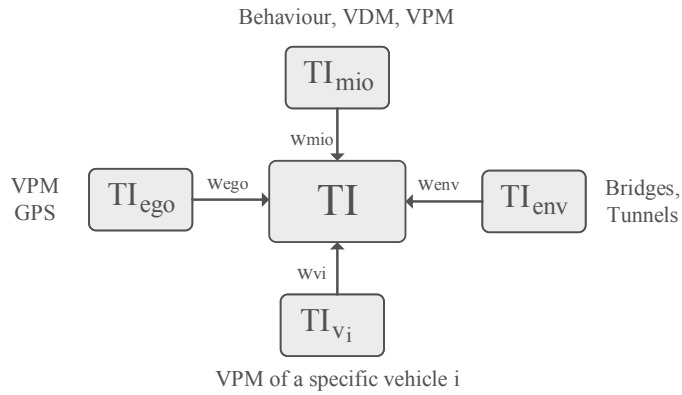


Figure G.2: Composition of TI.

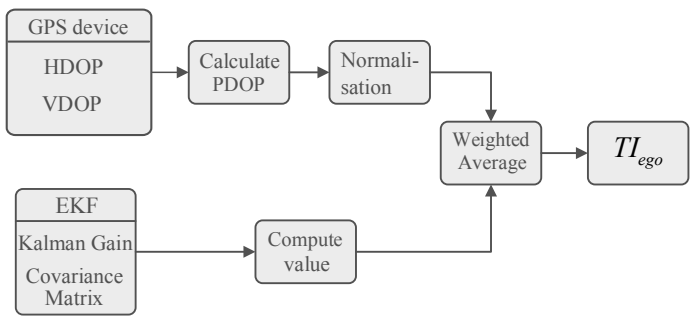


Figure G.3: Composition of TI_{ego}.

previously mentioned DOP values. These dimensionless values can be infinitely large [33]. Consequently, it is necessary to set a limit that indicates a highly imprecise measurement. This limit is set by the constant C, since a reliable geographical position is a prerequisite for cooperative driving. The limit of $C = 25$ is based on an experiment with disadvantageous weather conditions, namely intense rain and clouded sky, where platooning was still possible. This condition was set to $TI_{gps} = 0.5$. The Position DOP (PDOP) value is a combination of the former DOP values and used for further computation [33]. Equation G.2 provides the formulas for calculating TI_{ego}. The first step is to compute the PDOP value and normalise it with a maximum value, constant C. Next, the Kalman gain is evaluated by calculating the mean of the diagonal of K_k, which contains the factor of "trust" for each measurement. The overall TI of the ego vehicle representing the quality of the sensors is achieved by applying a weighted average of the indices describing the EKF and the GPS quality.

Each measurement described by the VPM has the same weight for calculating TI_{VPM} described in Equation G.2c. A higher weight for the position accuracy is not

necessary since the quality of the geographical position itself is already considered in TI_{gps}.

$$\text{PDOP} = \sqrt{\text{HDOP}^2 + \text{VDOP}^2} \tag{G.2a}$$

$$TI_{gps} = \begin{cases} 0 & \text{if PDOP} > C \\ 1 - \frac{\text{PDOP}}{C} & \text{otherwise} \end{cases} \tag{G.2b}$$

$$TI_{VPM} = \frac{\sum_{i=1}^{N} K_k(i,i)}{N} \tag{G.2c}$$

$$TI_{ego} = \frac{w_{gps} \cdot TI_{gps} + w_{VPM} \cdot TI_{VPM}}{w_{gps} + w_{VPM}} \tag{G.2d}$$

3.5.2 Trust Index TI_{mio}

TI_{mio} and TI_{v_i} are both combining trust in the measurement quality and trust in the behaviour of the other vehicles. A definition of correct or wrong behaviour is highly complex since one can consider many factors that help to define behaviour. A survey of the factors expressing the behaviour of a vehicle is shown in this section.

Vehicle behaviour can be described through observing historical data generated by the vehicle while driving. Figure G.4 illustrates the identified factors that help to conclude about the vehicular behaviour. The current velocity provides information about the cautiousness of the driver or automated system.

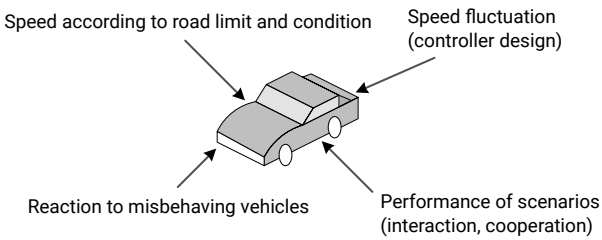


Figure G.4: Factors describing the behaviour of a vehicle.

The system of other vehicles might do something wrong or unexpected due to lack of perception. Vehicles have to react properly even if other road users make mistakes. The interaction with the surrounding traffic is important for cooperative driving. The main requirement for cooperative driving is using a common interaction protocol.

Speed Fluctuation. The stability of the speed of the MIO can be considered by analysing the speed profile. A condition for using the velocity profile of the MIO is a second sensor that is able to measure the speed of the preceding vehicle. A front radar is able to detect the distance and calculate the range rate to any obstacles in front. The range rate is defined in [34] as the rate of the change of the distance to the obstacle as defined in Equation G.3a.

$$\dot{R} = \frac{dR}{dt} \tag{G.3a}$$

$$v_{mio} = v_{ego} + \dot{R}, \tag{G.3b}$$

where

- R is range or distance to the detected obstacle,
- $\dot{R}$ is range rate, and
- v is the velocity of the MIO or ego vehicle.

Equation G.3b shows the calculation of the preceding vehicle's speed considering the speed of the ego vehicle and the range rate $\dot{R}$ of the MIO. The range rate in m/s provides information about the change of the distance in metres. The formula is only valid if the preceding vehicle is moving in the same direction as the ego vehicle. This requirement is fulfilled while the vehicles are driving on a straight road however it might be invalid in curves.

Interaction. This TI is GCDC specific, but can be applied to other interaction protocols as well. The communication between the vehicles during a scenario can be observed as long as they are within communication range of the ego vehicle. The highway scenario, scenario 1, of GCDC 2016 relies on the correct pairing of the vehicles in order to perform a safe merge from one lane to the other. The pairing procedure is described in the i-GAME deliverable 3.2 [29] and in [35]. The lack of interoperability can be a cause for wrong interactions. Observing a wrong pairing of the MIO, or another vehicle around the ego vehicle, can be used for consideration in both TI_{mio} and TI_{v_i}.

Another measure to validate the correct implementation of the interaction protocol is the observation of the Safe-to-Merge (STOM) message. Vehicles that have paired up correctly are making a gap relative to their pairing partner. As soon as the vehicle that is making a gap for a merging vehicle decides that the gap is large enough to merge, it sends out a STOM message. Observing the created gap when the STOM message is sent can be used to verify the trust in the vehicle when interacting with other vehicles. Future scenarios for cooperative and automated driving may provide more accurate measures to be considered in a TI.

Recommended Speed. The speed of the preceding vehicle indicates how safe it is driving. Considering the weather and the road conditions in combination with the knowledge about the current speed limit allows to conclude a recommended speed for safe driving.

Reaction to misbehaving vehicles. The reliability of the vehicle's control system is important and should be considered in the TI computation for each vehicle. The ego vehicle has to know, if the vehicle is not able to react to misbehaving vehicles properly, because the safety of the passengers in the vehicle itself and the surrounding vehicles might be at risk. Analysing the behaviour of all vehicles and evaluating the reaction of the surrounding vehicles needs significant computational power and different types of sensors are required.

Composition of TI_{mio}. Figure G.5 illustrates the measures that can be considered in TI_{mio}. *Interaction* can be used to signal how well the vehicle cooperates with others, e. g., if it pairs and sends STOMs correctly. The degree of the speed fluctuation is mapped using a predefined table and represented as the factor *Speed Fluctuation*. Taking into account the factors weather, road condition, and current speed limit, allows the computation of the recommended speed using inference. A comparison of this recommendation and the actual speed of the preceding vehicle is mapped to the partial TI *Speed*. The VDM and VPM described in Sections 3.2 and 3.3 provide information about the sensor quality of the vehicle in front. The reaction to misbehaving vehicles is expressed as the sixth factor to be considered for TI_{mio}.

Figure G.4 illustrates the factors that potentially influence the TI. Moreover, Figure G.5 shows the composition of TI_{mio}. Since the competition car was not able to

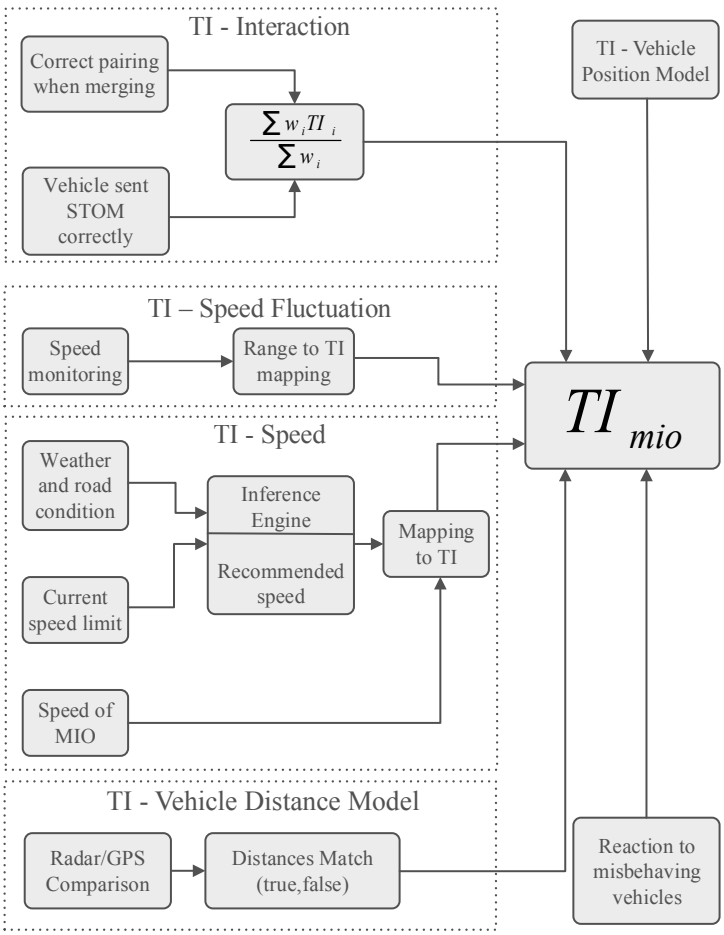


Figure G.5: Composition of TI_{mio}.

perceive all the mentioned measures, such as the current speed limit, road condition, and the observation of the reaction to misbehaving vehicles, this work focuses on the following four factors: the result of the VDM and VPM using an EKF, the speed fluctuation, and the performance when interacting with other vehicles.

3.5.3 Trust Index TI_{env}

The environment is categorised into static and dynamic. Roads, tunnels, bridges and the information about rural or urban areas are considered as static. Changes in the dynamic environment may be communication disturbances due to vehicles that block the radio waves e. g., a truck. Research about algorithms deciding what should be forwarded by the truck in order to enable the communication for all vehicles within a certain range is ongoing. Larsson proposes in [36] a performance centric forwarding algorithm.

Deciding whether being in an urban or rural environment can be made using several different methods. One method is to use a map of a navigation software. Another approach is using the PDOP value which is provided by the GPS device. By considering this precision information, it can be deduced if the vehicle is driving in a city, tunnel, under a bridge or on a highway. A less precise geographic position is common in cities, due to the reflection of the electromagnetic waves by buildings. An indicator that the vehicle is driving in a tunnel is the lack of position updates and high HDOP and VDOP values.

The knowledge of the environment, such as the information that the vehicle is in a city, is important. The situation awareness has to be increased when driving in cities, due to dense traffic, bicyclists, pedestrians and other vulnerable road users.

TI_{env} considers the PDOP value for indicating the trust in the environment. Moreover, TI_{env} is being stored in a database for prospective use. The mapping of a TI_{env} perceived in the past is accomplished using circles of a certain size, e. g., 5 metres.

3.5.4 Trust Index TI_{v_i}

This TI is similar to TI_{mio}. A system with enough resources for computation can observe and evaluate the interaction of the surrounding vehicles with each other. Moreover, more sensors and different types of sensors are needed in order to evaluate the precision of the other vehicle's information.

Another challenge is the calculation of a TI considering observations made by other vehicles. More sensors can increase the perception of the ego vehicle, but they can only be used to a certain extent. The system has to rely on the V2V information in case a vehicle cannot be perceived with the own sensors. For example, the distance between two vehicles can be compared with the calculated distance using the geographic position and the reported distance to the MIO.

3.6 V2V Perception

The perception range of the vehicle is strongly dependent on its level of automation. A fully autonomous vehicle has to make all decisions on its own as the driver is not in the control loop anymore. Thus, it needs to detect obstacles, other vehicles, pedestrians, traffic lights, speed limits, lanes markings and road conditions. The system used to evaluate the proposed concept has a limited perception and a longitudinal controller whereas the lateral control is performed manually. This restriction decreases the number of sensors needed to be able to drive automatically.

A map of the surrounding vehicles is essential for fulfilling the interaction of the scenarios described in [29]. The system needs to identify the vehicle in the front right/left as well as the vehicle behind on the right lane. To generate a map that fulfils this requirement with the available sensors, the system has to rely on the reported vehicle position provided via V2V communication, because only the position of the vehicle in front can be verified with the built-in radar. This approach has been chosen for the reason that the ego vehicle used for the TS has a limited perception. Vehicles with a more advanced perception should rely on their own sensors and consider the correctness of the information provided via V2V communication in the vehicle specific TI, TI_{v_i}.

The identification of the surrounding vehicle's position can be determined by considering the lane in which the vehicle is currently driving in or by computing the relative angles to the vehicles with the shortest distance. Both approaches are computed and further on combined with each other in order to achieve a more robust identification of the vehicle's relative position.

Lane. The CAM as well as the i-GAME Cooperative Lane Change messages (iCLCMs) contain a field describing the lane in which the vehicle is currently driving. The CAM lane is an integer in the range of -1 to 14. -1 refers to a position outside of the road, 0 is the hard shoulder and 1 is the outermost driving lane [28]. By taking the lane position of the vehicle into account, the algorithm is more robust against misidentified vehicles.

Angles. Considering only the relative angles of the vehicles and their distance provides a more robust identification of vehicles that are sending an incorrect or no lane ID. The downside of only using this algorithm for identifying the relative position of the surrounding vehicles is the risk of misclassification of vehicles when driving in a curve.

A method that combines both proposed algorithms takes the advantages of both. Vehicles in the GCDC 2016 mostly sent the correct lane ID and therefore, this algorithm is used as a base. In case a vehicle is identified at another position by the map that considers only the angles and the distance, this field is also updated in the overall map. This approach allows false positives, as it possible that one vehicle is identified at two places, for instance front-left and front.

Figure G.6 illustrates the ego vehicle and the relative position description of the surrounding vehicles. The first letter contains the information if the car is in the front of the vehicle or behind it, while the other letter indicates left or right. The blue areas indicate the angle range that is used for classifying the position of the vehicles.

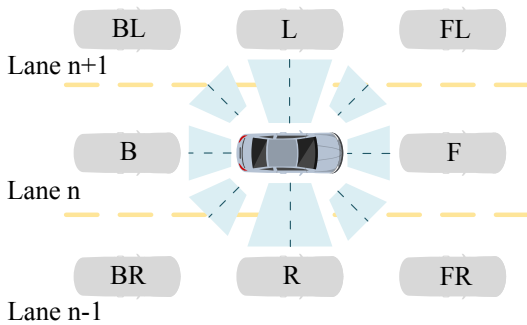


Figure G.6: Illustration of the map and the classification of the surrounding vehicles.

4 Experimental Investigations

The developed TS is evaluated with data from the highway scenario of the GCDC, because this scenario relies the most on the interaction with other vehicles via V2V communication. This section introduces the car, the GCDC highway scenario, and the parameters used for this experiment.

4.1 Car Setup

The proposed system has been implemented in a Volvo S60 provided by Volvo Car Corporation and tested and evaluated during and after the GCDC 2016. Java[4] was chosen as the primary programming language for realising the different functionalities of the car from team Halmstad. The communication between the modules of the system was achieved with Lightweight Communications and Marshalling (LCM)[5]. The benefit of using LCM is the ease of replaying log data and the real time observation of the exchanged data.

The competition car was equipped with antennas for the differential GPS and for the wireless communication. In addition to the antennas, there were two lights indicating whether the vehicle is in automated (green light) or manual (red light) mode.

The power is provided via a 12 V to 220 V power converter which is attached to an uninterruptible power supply. The communication between the devices is established with a wireless router. The V2X communication with the other vehicles and infrastructure is provided by the Alix System Board alix2d3[6]. To ensure a certain accuracy of the geographical position in the range of $+/-10$ cm in standstill, a differential GPS device is needed. The dSPACE MicroAutoBox, provided by Fengco Real Time Control AB, is used as the interface between the system and the car. Furthermore, the MicroAutoBox executes the low-level longitudinal controller. The

[4] https://java.com/en/
[5] https://lcm-proj.github.io/
[6] http://pcengines.ch/alix2d3.htm

system of the competition vehicle is executed on a computer, which is located in the front passenger seat.

4.2 Highway scenario

The highway scenario took place on the highway A270 between Eindhoven and Helmond in the Netherlands. Due to speed limitations of some participating vehicles, this scenario was split into two heats, one high speed and one low speed. In the high speed heat, the vehicles in the right lane, lane B, were driving with a speed of 60 km/h and the vehicles on the left lane, lane A, were driving at 80 km/h. In the low-speed heats the speed of the vehicles in lane A was 45 km/h and in lane B 40 km/h.

The highway scenario is split into four phases, pace making, parallel pairing, sequential pairing, and the merging phase. Figure G.7 illustrates phase III of the highway scenario, when the pairing between the vehicles is completed [29, 35].

I: Pace Making. At the beginning, the OPCs are bringing the vehicles into the right position. As soon as the vehicles are correctly positioned, a roadwork message is sent to all participants. The roadwork message implies a roadwork on a certain lane (lane A) and the reduction of all vehicles' speed to 40 km/h.

II: Parallel Pairing. A merge request triggers the so-called *B2A* pairing, which means that the vehicles on the right lane are setting their forward pair, the vehicle on the front-left, in the iCLCM message. This pairing is performed in parallel by all vehicles. Additionally, their forward partner acknowledges the pairing by setting them as its backward pair. When the *B2A* pairing is done, the vehicles in lane B create a gap so that their forward partner can merge in front of it.

III: Sequential Pairing. After a certain time, the lead vehicle on the left lane pairs up with the vehicle on the front-right and creates a gap. The front-right vehicle is identified as the MIO of the backward pair. As soon as the gap is large enough, the backward pair of the lead vehicle sends out a STOM message indicating that the gap is large enough for the lead vehicle to merge.

IV: Merging done. After merging, the lead vehicle adapts its parameters to the right platoon and the new lead vehicle on the left lane can start with the *A2B* pairing.

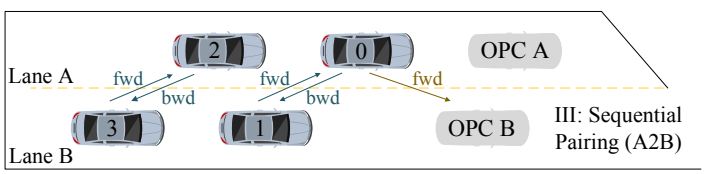


Figure G.7: Phase III of the GCDC highway scenario (reproduced from [29, p. 8]).

During the competition, one case was experienced, where the communication to all vehicles in front of two trucks driving next to each other on two adjacent lanes was blocked due to their physical properties. The only available information about the

vehicles in front is the MIO information transmitted by the trucks via the iCLCM message. With this information, the system of the vehicle has the knowledge that another vehicle is in front of its MIO with a certain distance. Other information, such as the preceding vehicle of the MIO of the truck cannot be gathered and thus an increased situation awareness is necessary for a correct evaluation of the situation.

4.3 Parameter Settings

The data gathered during the competition has been logged according to the requirements of the competition shown in [37] and with the use of *lcm-log*. The generated LCM logs can be played with the *lcm-logplayer*. Moreover, the logs were created for each module individually as a separate file containing only the received or transmitted data. Thus, it is possible to playback the log of the TS containing all information that the TS received. The use of recorded data was chosen for the reason that it is data from a real situation, which includes the interaction with other vehicles, sensor inaccuracies, and communication delays.

The weights of the TIs were chosen manually to demonstrate the general idea of the TS. All TIs, except TI_{ego}, have the same weight. TI_{ego} has a higher weight because it describes the trust in the ego vehicle. Equation G.4 shows the specific weights. It should be noted that the only appearance of TI_{v_i} is TI_{fwd} and has thus been weighted with the same weight as TI_{mio}.

$$w_{mio} = w_{v_i} = w_{env} = 3; \quad w_{ego} = 5; \tag{G.4}$$

Moreover, the adjusted weights for TI_{ego} described in Section 3.5.1 are listed in Equation G.5.

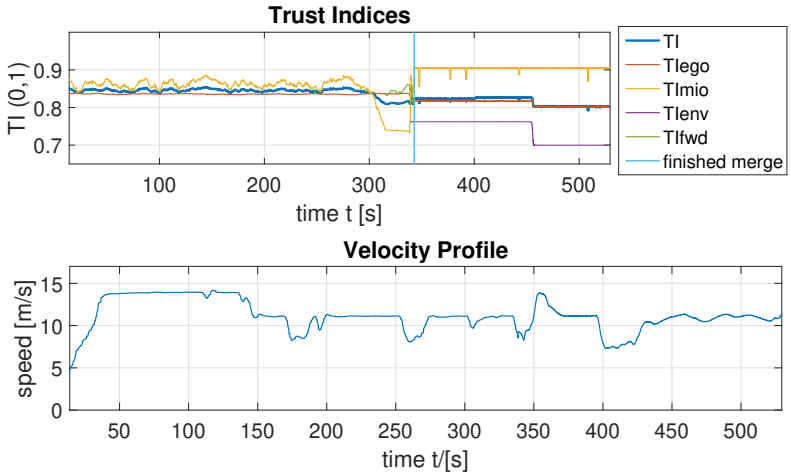
$$w_{gps} = 0.25; \quad w_{VPM} = 0.75; \tag{G.5}$$

5 Results

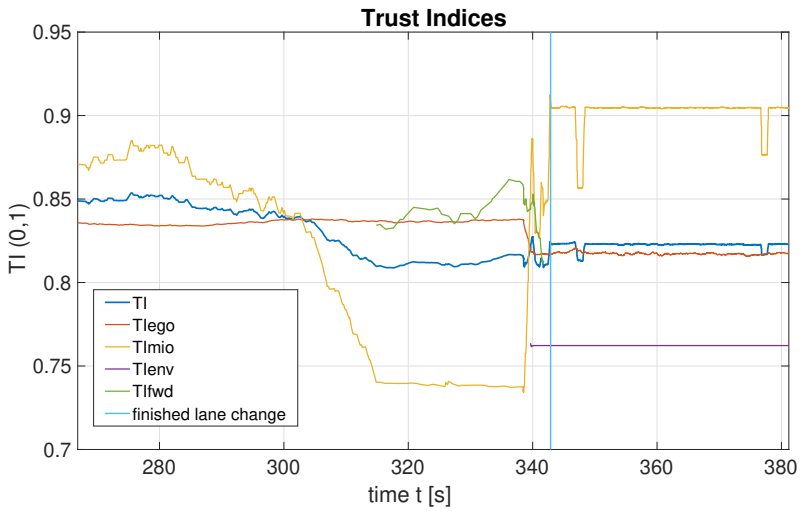
The results are shown as graphs illustrating the TI during different heats and different placements of the vehicle within the two platoons. The partial TIs are depicted in the same plots to show the composition and highlight the influences of each component. The generation of the TI has been simplified due to the limited performance of the system of the competition car. To ease visibility in the plots, the TIs have been filtered with a moving average filter with a twelve second window, which has been experimentally selected.

5.1 Vehicle merges into right lane

Figure G.8a illustrates the TI and the speed profile of the test vehicle during a low-speed heat. The speed profile is shown for the sake of completeness and omitted in further figures. The behaviour of TI_{ego} is stable with a value at around 0.84 until the vehicle merges. A level between 0.8 and 0.9 indicates for the decision making algorithm that the information of the ego, the other vehicles, and the environment is reliable and that the vehicle can decrease or keep its distance to the MIO. Whenever



(a) TI and speed profile.



(b) TI during the merge.

Figure G.8: Experiment I: Highway scenario starting from the left lane.

the TS does not have the necessary information to calculate a partial TI, it will omit it in the TI calculation. In a future implementation one may mark certain information sources as mandatory or optional. This way, the lack of information would be also considered in the TIs.

TI_{ego} decreases as soon as the TS has more information about its geographic position, which is shown by the appearance of TI_{env} that considers also the precision information (PDOP value) gathered from the GPS device. The late appearance of the PDOP value is caused by the GPS device that never provided this information in this

area (until $t = 340$ in Figure G.8a). It shows the influence of the knowledge about the environment on TI_{env} and TI_{ego}. TI_{env} is calculated by using the current PDOP value or a TI_{env} from the area close to the vehicle that was perceived in the past. Moreover, TI_{env} decreases at second 460 due to a change in the environment – trees covered the road and caused a decrease of the GPS precision.

Comparing TI_{mio} before and after the merge shows that there are different vehicles in front of the ego vehicle. It also indicates that the MIO after the merge is more reliable compared to the preceding vehicle before the merge. The vertical blue line indicates the time when the merge to the right lane has finished. It can also be seen that TI_{mio} is decreasing while the merge is being performed because the system cannot detect the correct MIO during that phase and thus the radar match is negative.

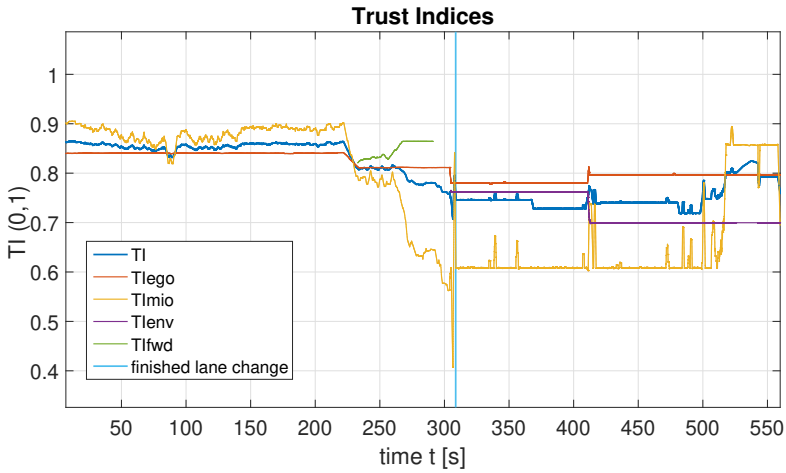
The forward partner is only set within the *B2A* and the merging phase. TI_{fwd} describes the trust in the forward partner and is thus only calculated within these phases. This TI originates from TI_{v_i}. An extract of the TI during the merge from second 250 to 380 can be found in Figure G.8b. It shows TI_{fwd} in more detail and it also illustrates the behaviour of TI_{mio} during the merge.

The decreased TI during the merge can be used to inform the decision making module that it has to decrease its trust into the current situation. However, the lane change has to be treated as a special case, since the car has to keep its speed in order to not interfere with the interaction protocol. Furthermore, the trust in the new preceding vehicle is stable with a value of around 0.9, because the match between radar distance and geographical position is more accurate than with the previous MIO. This information can be used to maintain safety while decreasing the time headway or the distance to the preceding vehicle when this vehicle provides reliable data, i.e. has a high TI.

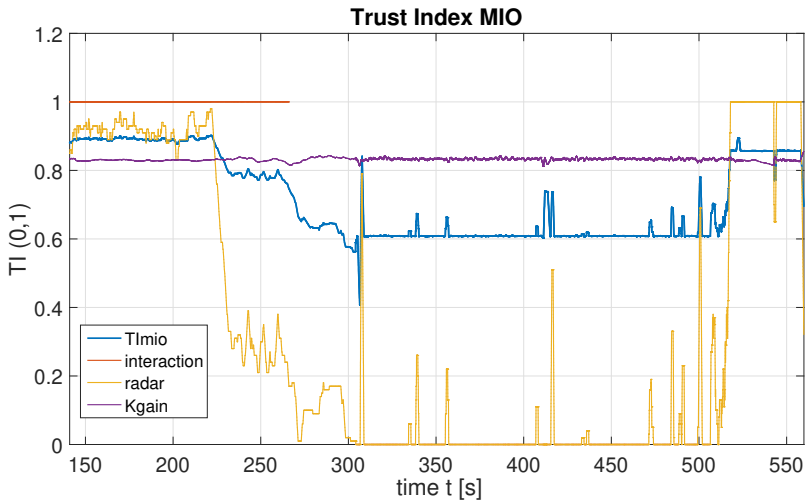
5.2 Making gap for vehicle to merge

In the highway scenario, when the vehicle starts in the right lane, it does not need to change the lane. It creates a gap with respect to its forward partner on the left lane and sends out a STOM message as soon as the gap is large enough for the forward partner to merge.

Figure G.9 depicts the composition and behaviour of the TI and TI_{mio}. TI_{ego} shown in Figure G.9a is also stable and changes when the PDOP value occurs. The behaviour of TI_{fwd} is similar to the one presented in Figure G.8a. First, TI_{fwd} is lower and increases with the experience gathered about the vehicle. It shows also the same behaviour during the merge. As soon as the vehicle, in this case the forward partner, changes the lane, TI_{mio} is decreasing significantly due to the change of the ego vehicle's preceding car. Furthermore, the new MIO has a lower TI compared to the previous one. This is caused by a continuous misidentification of the MIO. Due to this misidentification, the observed values with the radar and the reported data via V2V communication do not match, the TI decreases. The reason for this continuous misidentification is a low accuracy of the received geographical position. It was experienced that the new preceding vehicle had provided delayed position data and thus the system could not identify it correctly.



(a) Behaviour of the TI



(b) Composition of TI_{mio}.

Figure G.9: Experiment II: Highway scenario starting from right lane.

Figure G.9b depicts the partial TIs used to calculate TI_{mio}. For a better illustration of the TIs, the same moving average filter has been applied. This filter is the reason why the TI of the radar is not binary. The forward partner becomes the new MIO after the merge. Since TI_{fwd} only uses the VPM for calculating this index, TI_{mio} does not necessarily match with TI_{fwd}. The behaviour of TI_{mio} is strongly influenced by the result of the radar match. After the merge, from second 300 to 510, the measured radar distance differs from the distance using the geographical position. The evaluation of the radar match is performed within the VDM of the system of the car as described in Section 3.2.

The identification of measurement mismatches by comparing the own sensor data of the ego vehicle with the reported data is important for situation awareness. The implemented controller did not have any problems with following this vehicle in a platoon because the system relies, for safety reasons, in such cases on the radar information. The decreased TI_{mio} tells the other modules of the system that the reported data can not be fully trusted.

The graph shows that the trust in the new MIO after the merge is about 0.6. Comparing the overall TI of the system highlights that the situation awareness has to be increased. A possible reaction to this behaviour can be an increased time headway or distance to the preceding vehicle. An overall TI of less than 0.5 means that the TS has evidence for not trusting this vehicle and the gap to the preceding vehicle has to be increased.

5.3 Unreliable Geographical Position

The proposed system has to rely to a certain extent on the geographical position provided by the other vehicles because the system is only able to verify the position of the preceding vehicle and the VPM is not capable of improving the position of highly inaccurate position data or unreliable data. One case was experienced in the course of the GCDC where the intended forward partner provided unreliable position information.

The introduced TS is able to improve the geographical position of a vehicle by considering the inertial sensor information, but it cannot improve the position of highly inaccurate measurements that provide changes in the position from $+/-80$ metres. Figure G.10 illustrates such a situation. The position of the competition car is plotted in blue as a reference. The red curve shows the position of the forward partner. As can be seen, the geographical position of the other participant's vehicle is fluctuating to a large extent. Unfortunately the source of the disturbance is not known and could not be investigated. From experience the characteristics of the disturbance indicate that it may be caused by synchronization problems with the Real Time Kinematic (RTK) base station.

Vehicles which provide such an unreliable position cannot be identified with the current implementation of the proposed TS. This is caused by the competition car's limited perception with its own sensors. For computational reasons, only vehicles that behave properly to a certain extent are monitored, i.e. vehicles with large fluctuations in position can not be followed with the use of the VDM.

6 Conclusion

This paper presents a Trust System (TS) capable of perceiving the local environment and generate a Trust Index (TI) describing the overall reliability of both the on-board vehicle sensor data as well as the data received through V2V communication. The TS perceives the vehicle's local environment and generates a TI indicating the system's level of trust in the sensor readings and their reliability at any given time instant. The TI is within the range 0 to 1, and takes several factors into account e.g. the

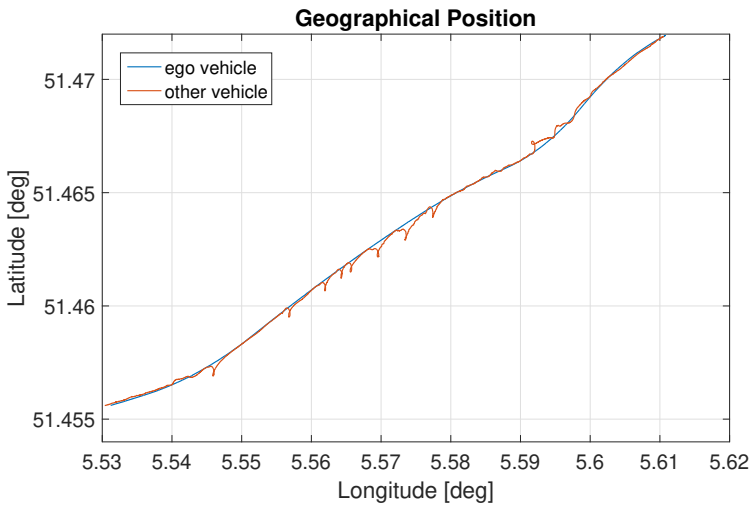


Figure G.10: Example of a vehicle reporting an unreliable/inaccurate geographical position.

environment itself, the ego vehicle, the other vehicles, and in particular the preceding vehicle. The TI is broadcasted to the modules of the vehicle's system to inform them about the current situation and allow them to consider it when taking actions.

The evaluation of the sensor accuracy of the ego vehicle and the other vehicles is performed with the vehicle distance model (VDM) and the vehicle position model (VPM). The VDM describes the relation between the distance measured with a radar and the distance based on the provided geographical position. A kinematic model of the vehicle in combination with an EKF has been implemented in the VPM. The TS generates the TI based on various factors that can influence the situation awareness.

The results of the TS are illustrated and discussed. The behaviour of the TI in various situations shows the correct identification of situations, where the preceding vehicle has a lower reliability and thus a lower TI is assigned to the vehicle. A discussion about the possible influences of the proposed TS in decision making is also given and indicates that, to maintain safety while the TI is low, the TS can suggest to increase the time headway to the preceding vehicle or reduce the own vehicle's speed. The overall TI consists of several partial TIs, future work may include adaptive weighing of these TIs in different situations. This can be achieved through further testing in various situations.

Acknowledgements. The authors would like to thank the i-GAME project for preparing and organising GCDC 2016, the Swedish CoAct project and institutions for supporting us organisationally and financially, and our sponsors including Fengco, TASS International, and Volvo Cars.

Bibliography

[1] World Health Organization (WHO), *Global Status Report on Road Safety 2015*. Geneva: WHO Press, 2015.

[2] A. Davila and M. Nombela, "Platooning - safe and eco-friendly mobility." *SAE Technical Paper 2012-01-0488*, 2012.

[3] J. Dokic, B. Müller, and G. Meyer, "European roadmap smart systems for automated driving," *European Technology Platform on Smart Systems Integration*, 2015.

[4] R. N. Charette, "This Car Runs on Code," 2009. [Online]. Available: http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code

[5] T. Bijlsma, S. de Kievit, J. van de Sluis, E. van Nunen, I. Passchier, and E. Luiijf, "Security challenges for cooperative and interconnected mobility systems," in *International Workshop on Critical Information Infrastructures Security*. Springer, 2013, pp. 1–15.

[6] S.-W. Kim, Z. J. Chong, B. Qin, X. Shen, Z. Cheng, W. Liu, and M. H. Ang, "Cooperative perception for autonomous vehicle control on the road: Motivation and experimental results," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5059–5066, 2013.

[7] S. W. Kim, B. Qin, Z. J. Chong, X. Shen, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus, "Multivehicle Cooperative Driving Using Cooperative Perception: Design and Experimental Validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 663–680, 2015.

[8] J. Ploeg, B. T. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, no. June 2016, pp. 260–265, 2011.

[9] L. Chen and C. Englund, "Cooperative Intersection Management: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 570–586, 2016.

[10] R. Kianfar, B. Augusto, A. Ebadighajari, U. Hakeem, J. Nilsson *et al.*, "Design and Experimental Validation of a Cooperative Driving System in the Grand Cooperative Driving Challenge," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 3, pp. 994–1007, sep 2012.

[11] K. Lidström, K. Sjöberg, U. Holmberg, J. Andersson, F. Bergh, M. Bjäde, and S. Mak, "A Modular CACC System Integration and Design," 2012.

[12] M. Röckl, J. Gacnik, and J. Schomerus, "Integration of car-2-car communication as a virtual sensor in automotive sensor fusion for advanced driver assistance systems," in *FISITA 2008*, A. of German Engineers (VDI), Ed. Springer Automotive Media, 2008. [Online]. Available: http://elib.dlr.de/55254/

[13] M. Aramrattana, T. Larsson, J. Jansson, and C. Englund, "Dimensions of cooperative driving, its and automation," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 144–149.

[14] P. Wex, J. Breuer, A. Held, T. Leinmuller, and L. Delgrossi, "Trust Issues for Vehicular Ad Hoc Networks," *VTC Spring 2008 - IEEE Vehicular Technology Conference*, pp. 2800–2804, 2008.

[15] J. Zhang, "A survey on trust management for VANETs," *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pp. 105–112, 2011.

[16] U. F. Minhas, J. Zhang, T. Tran, and R. Cohen, "Towards expanded trust management for agents in vehicular ad-hoc networks," *International Journal of Computational Intelligence Theory and Practice (IJCITP)*, vol. 5, no. 1, 2010.

[17] M. Raya, P. Papadimitratos, V. D. Gligor, and J. P. Hubaux, "On Data-Centric Trust Establishment in Ephemeral Ad Hoc Networks," *IEEE INFOCOM 2008*, pp. 1912–1920, 2008.

[18] F. Dötzer, L. Fischer, and P. Magiera, "VARS: A vehicle ad-hoc network reputation system," *Proceedings - 6th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, WoWMoM 2005*, pp. 454–456, 2005.

[19] P. Agarwal and N. Bhardwaj, "A Review on Trust Model in Vehicular Ad Hoc Network," *International Journal of Grid and Distributed Computing*, vol. 9, no. 4, pp. 325–334, 2016.

[20] A. Bhargava, S. Verma, and B. K. Chaurasia, "Kalman filter for trust estimation in VANETs," in *2015 IEEE UP Section Conference on Electrical Computer and Electronics, UPCON 2015*, 2015.

[21] H. Aras, C. Beckstein, S. Buchegger, P. Dittrich, T. Hubauer, F. Klan, B. König-Ries, and W. Ouri, "08421 working group: Uncertainty and trust," in *Uncertainty Management in Information Systems*, ser. Dagstuhl Seminar Proceedings, C. Koch, B. König-Ries, V. Markl, and M. van Keulen, Eds., no. 08421. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2009/1938

[22] ETSI (European Telecommunications Standards Institute), "Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 3: Network architecture," *ETSI TS 102 636-3*, vol. 1, pp. 1–23, 2010.

[23] L. Chen and C. Englund, "Cooperative ITS - EU standards to accelerate cooperative mobility," in *The 3rd International Conference on Connected Vehicles & Expo (ICCVE 2014)*, 2014, pp. 681–686.

[24] ETSI (European Telecommunications Standards Institute), "Vehicular Communications ; Basic Set of Applications ; Part 2 : Specification of Cooperative," *ETSI TS 102 637-2*, vol. 1, pp. 1–18, 2011.

[25] D. MILBERT, "Dilution of precision revisited," *Navigation*, vol. 55, no. 1, pp. 67–81, 2008. [Online]. Available: http://dx.doi.org/10.1002/j.2161-4296.2008.tb00419.x

[26] E. Kaplan and C. Hegarty, *Understanding GPS: Principles and Applications*. Artech House, 2005.

[27] R. Yarlagadda, I. Ali, N. Al-Dhahir, and J. Hershey, "Gps gdop metric," *IEE Proceedings - Radar, Sonar and Navigation*, vol. 147, no. 5, pp. 259–264, Oct 2000.

[28] ETSI (European Telecommunications Standards Institute), "Intelligent Transport Systems (ITS); Users and applications requirements; Part 2: Applications and facilities layer common data dictionary," *ETSI TS 102 894-2*, vol. 1, pp. 1–78, 2013.

[29] J. van de Sluis, L. Chen, and L. Garcia-Sol, "DEL_i-GAME_D3.2 Proposal for extended message set for supervised automated driving," i-game gcdc, Tech. Rep., 2015.

[30] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, p. 35, 1960.

[31] R. Faragher, "Understanding the basis of the kalman filter via a simple and intuitive derivation," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 128–132, 2012.

[32] N. Magnusson and T. Odenman, "Improving absolute position estimates of an automotive vehicle using GPS in sensor fusion," Master's thesis, Chalmers University of Technology, 2012.

[33] R. B. Langley, "Dilution of Precision," *GPS World*, vol. 10, no. May, pp. 52–59, 1999.

[34] B. S. Bourgeois, "Using Range and Range Rate for Relative Navigation," Naval Research Laboratory Marine, Mississippi, Tech. Rep., 2007.

[35] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff, "The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 146–152, August 2016.

[36] M. Larsson, *Methods to Improve V2V Communications in Platoons of Heavy Duty Vehicles*. Halmstad University Press, 2016.

[37] H. Salunkhe, H. H. van Huysduynen, B. Nijssen, and J. Terken, "DEL_i-GAME_-D7.1 GCDC judging criteria and their evaluation (Final)," i-game gcdc, Tech. Rep., 2016.

Chapter H

V2C: A Trust-Based Vehicle to Cloud Anomaly Detection Framework for Automotive Systems

Adapted version that appeared in ARES 2021

T. Rosenstatter, T. Olovsson, M. Almgren

Abstract. Vehicles have become connected in many ways. They communicate with the cloud and will use Vehicle-to-Everything (V2X) communication to exchange warning messages and perform cooperative actions such as platooning. Vehicles have already been attacked and will become even more attractive targets due to their increasing connectivity, the amount of data they produce and their importance to our society. It is therefore crucial to provide cyber security measures to prevent and limit the impact of attacks.
As it is problematic for a vehicle to reliably assess its own state when it is compromised, we investigate how vehicle trust can be used to identify compromised vehicles and how fleet-wide attacks can be detected at an early stage using cloud data. In our proposed V2C Anomaly Detection framework, peer vehicles assess each other based on their perceived behavior in traffic and V2X-enabled interactions, and upload these assessments to the cloud for analysis. This framework consists of four modules. For each module we define functional demands, interfaces and evaluate solutions proposed in literature allowing manufacturers and fleet owners to choose appropriate techniques. We detail attack scenarios where this type of framework is particularly useful in detecting and identifying potential attacks and failing software and hardware. Furthermore, we describe what basic vehicle data the cloud analysis can be based upon.

V2C: A Trust-Based Vehicle to Cloud Anomaly Detection Framework for Automotive Systems

1 Introduction

To increase traffic safety and efficiency, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication is needed to enable the sensing of objects and entities that are out of the vehicle's sight; to allow cooperation between vehicles to increase efficiency such as in platooning [1] and to receive warnings about road conditions, e. g., slippery roads, traffic accidents and road works ahead. In addition, most vehicles also have cellular access to the Internet to provide comfort functions such as remote unlock/lock and to receive essential software updates over the air.

The feasibility of cyber attacks against vehicles that can potentially lead to catastrophic events was demonstrated already in 2011. Checkoway et al. [2] analyzed a modern vehicle for security vulnerabilities and identified several potential attacks requiring physical access, but also attacks possible through short-range wireless connectivity, e. g., Bluetooth, and long-range wireless connectivity such as cellular networks. In 2015 Miller and Valasek [3] demonstrated a remote exploit through the cellular network allowing them to remotely control a vehicle including safety-critical functions. A more recent report from 2020 revealed several vulnerabilities allowing remote control of Mercedes-Benz vehicles [4]. Remote attacks pose a significantly higher risk to the safety of road users as they do not require physical access to vehicles and can be performed from anywhere without leaving significant traces.

Security measures have been investigated and proposed for various parts of the automotive system, ranging from intrusion detection for the in-vehicle network (e. g., [5, 6]), secure communication for all networked communications and secure boot for Electronic Control Units, ECUs (e. g., [7]). These methods are important to protect and secure an individual vehicle and its occupants, however, they are not foolproof nor enough to identify wide-spread attacks and anomalies on a fleet level.

Motivation. Analyzing all available data in the cloud when an event from a single vehicle is received is not feasible due to the potentially large number of vehicles in the fleet and the associated computational costs. In general, it is also problematic for a system, i.e., the vehicle such as a passenger car, truck or bus, to reliably assess its own state when it is compromised. Therefore, a framework that allows finding anomalies and intrusions in automotive systems in its entirety is needed. The establishment of *trust* between vehicles interacting via V2V communication is of particular interest as it enables the assessment of a vehicle's behavior, especially in safety-critical situations, not just by the vehicle itself but also by the surrounding vehicles.

Contributions. In this paper, we present *V2C Anomaly Detection*, an overall framework that combines the detection of anomalies by evaluating V2V interactions using

trust scores with the analysis in the cloud. We divide this framework in modules and for each module, we survey existing solutions, investigate how to implement them, propose modifications when necessary and evaluate them. During the design we put special focus on scalability of the framework and the feasibility to detect attacks and abnormal behavior. We further argue that a framework like *V2C Anomaly Detection* is necessary and complements existing security controls and internal Intrusion Detection Systems (IDSs), as it is based on vehicles independently evaluating each other rather than only evaluating their own internal state.

After giving an overview of the system design and attacks in Section 2, we present related work in Section 3, review existing methods for each module in Section 4, and evaluate suitable methods and, when necessary, suggest modifications in Section 5. Ultimately, we show how to apply and discuss the *V2C Anomaly Detection* framework based on a detailed use case in Section 6 and then conclude the paper in Section 7.

2 Overview

Our goal is to utilize techniques in both individual vehicles and in the cloud in order to identify compromised vehicles. Security systems inside these compromised or malfunctioning vehicles may not be able to detect the misbehavior themselves and therefore a peer evaluation of independent systems is necessary. In addition, the analysis in the cloud with access to data about each vehicle in a fleet allows an early detection of large-scale attacks.

We first discuss the security threats and attack scenarios that the proposed framework should be able to detect in Section 2.1. An overview of the structure and the modules of the *V2C Anomaly Detection* framework using V2V and cloud data is presented in Section 2.2. Thereafter, an adversary model and assumptions are presented in Section 2.3.

2.1 Attack Scenarios

The following three attack scenarios highlight the need for trust-based anomaly detection in the cloud: (1) when firmware is successfully modified without triggering an alert from other monitoring software in the vehicle; (2) when hardware/software failures occur that cause an incorrect perception or behavior; and (3) when failed updates or faulty software cause functional disturbances. These scenarios and the examples listed in Table H.1 emphasize that the *V2C Anomaly Detection* framework focuses not only on detecting intrusions caused by unauthorized entities, it also aims at detecting behavioral changes caused by intentional and unintentional, yet authorized, modifications as well as random faults. These scenarios will be further used in Section 5.1 to evaluate the applicability of proposed trust and reputation models.

Scenario 1 – Unauthorized Firmware manipulation. Someone such as the owner, an employee at a workshop or a remote attacker, performs unauthorized modifications of the firmware in order to (i) suppress relaying of messages in the Vehicular Ad-hoc NETwork, VANET (blackhole attack), (ii) modify or falsify warning messages for

Table H.1: Examples for the identified scenarios.

Ex. #	Description
Scenario 1 – Unauthorized firmware manipulation	
Ex.1	Manipulation of the firmware such that the owner is able to send traffic congestion warning messages at will to reduce traffic on a desired road segment.
Ex.2	Manipulation of the firmware such that the automated vehicle drives faster than the current speed limit.
Ex.3	Manipulation of the firmware such that an attacker can disrupt traffic by suppressing relaying of messages, spoofing warning messages or flooding the Vehicular Ad-hoc NETwork (VANET) with erroneous messages.
Scenario 2 – HW/SW failures	
Ex.4	A lidar sensor or camera is experiencing a fault and the vehicle is thus not able to perceive its environment properly.
Scenario 3 – Legitimate SW/HW updates	
Ex.5	After a legitimate firmware update, the automated vehicle drives too fast in certain road conditions, e. g., when the road is slippery, since the vehicle perceives the current driving conditions incorrectly.
Ex.6	A defect hardware component is replaced in an authorized workshop and causes compatibility problems resulting in a misbehavior while driving.
Ex.7	The machine learning algorithm for identifying traffic signs has been updated and now causes a misclassification of speed limit signs. The possibility of attacks exploiting machine learning algorithms for the identification of traffic signs has also been demonstrated by Sitawarin et al. [8]. Thus, an update of such systems may even open new attack vectors.

events such as traffic accidents, traffic jams and other road conditions (masquerading attack) or (iii) interact with other vehicles during cooperative scenarios, e. g., platooning, in a selfish or malicious way to cause disruption of the traffic flow or even an accident.

Scenario 2 – HW/SW failures. Hardware or software faults occur and cause the vehicle to react improperly to the situation.

Scenario 3 – Legitimate SW/HW updates. The vehicle manufacturer pushes an over-the-air update for one of the computing units, ECUs, in a vehicle or an authorized workshop replaces hardware and causes a degraded or unintended functionality due to the complexity of an automotive system and lack of sufficient testing. Examples of

such unintended functionality are inaccurate or wrong sensor readings and changes in the behavior when interacting with other vehicles.

2.2 System Design

The goal of our *V2C Anomaly Detection* framework is to identify anomalous and malicious behavior in a fleet by having vehicles evaluate each other. As a result, we identified four tasks: (i) individually assess vehicle trust which consequently has to indicate anomalous behavior; (ii) combine these trust evaluations; (iii) detect a change in the combined trust evaluations showing that the vehicle's behavior negatively changed; and (iv) analyzing cloud data to identify similar patterns in the fleet. Each task is assigned to a module as shown in Figure H.1. *Module 1* evaluates its own and the behavior of other vehicles based on V2V data and the cooperation with them. Each vehicle performs this individual peer evaluation which results in a *trust score* and uploads the average of the trust scores for a certain period, e. g., per day, for each evaluated vehicle to the cloud. These trust scores reported by the vehicle fleet are combined to one trust score per vehicle in *module 2*. Combining the trust scores in the cloud has the advantages that no additional computation for combining trust scores is required by the vehicles and that no additional messages need to be exchanged in the VANET. *Module 3* observes the trust scores over time and raises an alert when the trust score of a specific vehicle decreases, indicating that the behavior of the vehicle has negatively changed. *Module 4* then analyzes the available data about the affected vehicle in the cloud to find the cause and allow an early detection in the fleet.

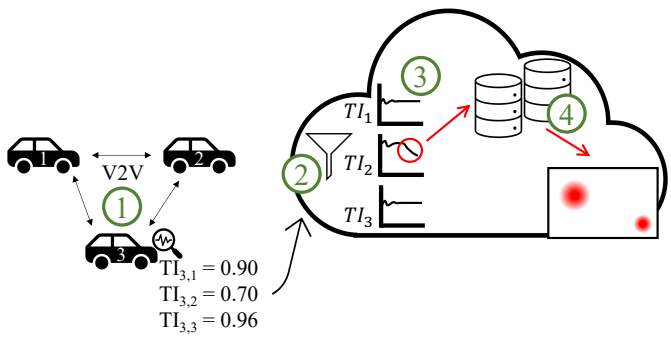


Figure H.1: Structure of *V2C Anomaly Detection*.

Since this is a framework providing flexibility for selecting and adapting suitable techniques for each module, we provide examples and show how the *V2C Anomaly Detection* framework can be used. Figure H.2 shows the required information as input, the output each module produces and links to the respective sections where this is further discussed.

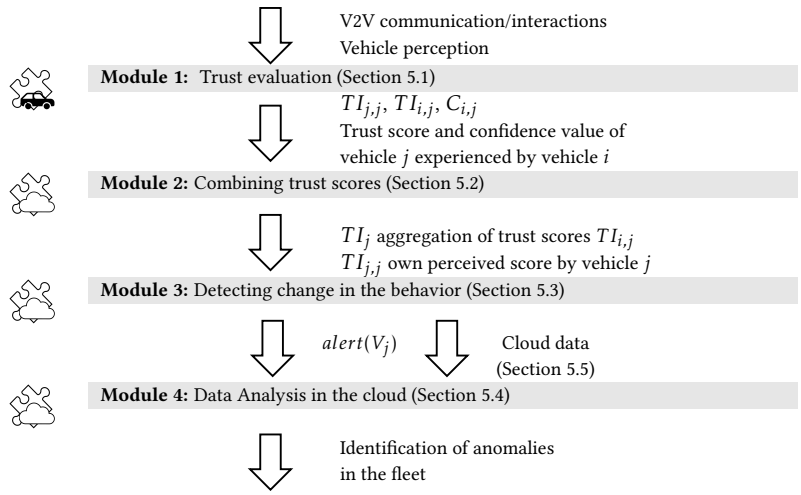


Figure H.2: Modules of the *V2C Anomaly Detection* framework including their inputs and outputs they produce.

2.3 Adversary Model and Assumptions

Attackers may gain complete control of a vehicle, e. g., through attack scenarios described in [9], allowing them to remotely control it or alter the firmware of important ECUs to perform attacks as mentioned in Section 2.1. It is reasonable to assume that many attacks can be perform stealthy and will only be detected through the behavior of the vehicle.

We assume that the majority of vehicles are honest and report correct trust scores, however, a minority of compromised vehicles may collude. The trust score calculated from the data and behavior involving V2V communication can also be correlated to the data about the vehicles located in the cloud.

3 Related Work

In this section we list related work in regards to this framework, while Section 4 reviews individual methods for each module. The introduction of VANETs and subsequent applications introduce additional security and safety challenges. Cryptographic solutions build the base for secure communication to provide confidentiality, integrity and availability. Nevertheless, dishonest or compromised vehicles need to be considered as well. VANET-specific attacks include spoofing locations of vehicles, sending altered or dropping warning messages and dropping all messages.

Trust and reputation models for evaluating vehicles based on V2V interactions have been proposed in literature [10] to identify vehicles that provide false or incorrect information to others and thus risking the safety of the passengers. Proposed solutions focus on how to evaluate the correctness of reported events, how to verify the correctness of sensor data, how to assess the interaction during cooperative

events such as platooning and/or how to distribute this knowledge between the vehicles. The resulting evaluation, a trust or reputation score, is used for decision-making by the automated vehicle. Examples are decisions about whether to trust reported warnings, e. g., road accident ahead, received from a particular vehicle. Guo et al. [11] propose an approach that verifies information received via V2V in form of an IDS whose results are further used for decision-making. Other solutions on collaborative intrusion detection [12, 13] consider only packet headers and parameters, such as packet drop rate and transfer delay, and investigate ways to exchange this information between neighboring vehicles.

To the best of our knowledge there is no similar work to the *V2C Anomaly Detection* framework, which deals with how trust scores can be utilized in a more holistic setting in order to detect large-scale attacks and anomalies by uploading and performing cloud-based analyses.

4 Existing Methods

The following sections survey existing methods relevant for each module. In Section 5 we discuss why and how these methods can be applied and motivations when modifications are necessary.

4.1 Trust and Reputation Models

The main task for module 1 is to evaluate vehicle behavior based on V2V data received and ways to assess cooperative tasks performed together. We use a recent survey on trust solutions for VANETs, namely Hussain et al. [10], as a base and further perform a search on *Google Scholar* to find additional trust and reputation-based models. We briefly present a diverse set of solutions to give an overview of methods found in literature, but refer to Hussain et al. [10] for a more complete overview of solutions dealing with trust or reputation in VANETs.

The reputation model presented by Engoulou et al. [14] verifies the vehicle id, width and length of the vehicle, driving direction, location, speed, acceleration, transmission rate and message frequency. Each of these parameters is either correct (1) or false (0) and the resulting sum is the proposed reputation score. This model lacks the evaluation of the other vehicles' cooperative behavior, for example, if the vehicles provide correct warning messages and how they behave during cooperative scenarios such as platooning.

Soleymani et al. [15] make use of fuzzy logic to model trust. Three modules, such as an experience module which evaluates the interactions, a plausibility module which verifies the location of the sender, and an accuracy level module, result in one of the three levels: *high*, *medium* or *low*. A fuzzy inference engine further defines the combinations in which the trust level is *acceptable* and *not-acceptable*. The approach of using fuzzy logic is interesting, however, for our proposed anomaly detection framework we require a more detailed representation of trust/reputation.

A trust system focusing on modeling the trust in the surrounding vehicles as well as the ego vehicle is presented by Rosenstatter and Englund [16]. The authors identify

trust evaluation criteria for the own (ego) vehicle, all surrounding vehicles including more specific criteria for the vehicle in front, as this vehicle can be additionally verified with the vehicle's front sensors. Unlike other proposed solutions, this system has been evaluated with data from a real environment consisting of several Vehicle-to-Everything (V2X)-enabled vehicles. In addition, the presented trust model is flexible for adaptations, which is also shown by the authors' detailed discussions of factors that can be considered for evaluating the behavior of a vehicle.

Bißmeyer et al. [17] propose a trust score based on plausibility checks, e. g., reported location, using particle filters. An aging factor is introduced to define ratio of the impact of a new trust score compared to the past trust scores. The authors also propose to apply an additional particle filter to the ego vehicle to verify the trust in the own system as the ego vehicle acts as the reference for evaluating the other vehicles. An extension of the proposed model would be necessary to cover also an evaluation of the behavior of other vehicles during cooperative scenarios such as platooning.

In Section 5.1 we return to trust models and give more details on the demands on the trust models and how they can be applied in the context of the *V2C Anomaly Detection* framework.

4.2 Combining Trust Scores

The combination of knowledge, i. e., the trust scores $TI_{i,j}$, reported by individual vehicles can be carried out in various ways by module 2. Overall, there are two distinct concepts for combining trust scores, (i) calculating the mean, e. g., weighted, arithmetic or geometric, and (ii) using Dempster-Shafer theory (DST) of evidence [18], more specifically, Dempster's rule of combination.

Arithmetic and Geometric Mean. The trust scores reported by each vehicle need to be combined in such a way that it compensates for a minority of dishonest vehicles. The arithmetic mean is generally used for data with no significant outliers whereas the geometric mean is used when the difference between the data points is logarithmic.

The arithmetic mean, for instance, is used by Engoulou et al. [14] to calculate the indirect trust, an aggregation of the trust scores of vehicle j received from other vehicles. Furthermore, the authors use a weighted average for combining the local trust, calculated by the own system for vehicle j and the indirect trust. Rosenstatter and Englund [16] also use the weighted average for the calculation of the combined trust score reflecting the current situation.

Halabi and Zulkernine [19] propose a cooperative game model for preventing malicious vehicles from entering a vehicle coalition. The authors argue for using the product of the trust scores ($TI_{i,j}$) when computing the trust in a vehicle group to increase the impact of low trust scores. A model using the geometric mean is also presented by Kerrache et al. [20]. The described trust model evaluates only binary events and combines the received trust ratings using geometric mean, however, the authors do not provide any reasoning for choosing the geometric mean over the arithmetic mean.

Dempster-Shafer Theory. DST is used in situations which require the combination of evidence reported by several (unreliable) observers. Chen and Venkataramanan [21] present how DST can be applied in the context of intrusion detection in VANETs. The authors argue that Bayesian inference is less suited for such a use as it requires the complete knowledge of prior probabilities, which often need to be estimated in practice. DST, however, handles the lack of a complete probabilistic model by introducing belief and plausibility instead of probabilities. Chen and Venkataramanan highlight the difference with the following example: Node A is trustworthy with a probability of 0.8 respectively untrustworthy with probability 0.2 and reports that node S is trustworthy. In the event that A is indeed trustworthy, the claim that S is trustworthy is accurate, but A being not trustworthy, does not automatically imply that A is inaccurate – it says that the claim of A has 0.8 degrees of belief for S being trustworthy and 0 (not 0.2) degrees of belief that S is untrustworthy [21].

Combining trust or reputation scores reported by several vehicles using DST has been proposed in several models. For instance, Zhang et al. [22] use the computed reputation value per vehicle as degree of belief for either trusting or distrusting a vehicle. A unit in the cloud, a so-called trust authority, combines these reports according to Dempster's rule of combination.

A similar approach as proposed by Chen and Venkataramanan, and Zhang et al. can potentially be applied to the trust score. The trust score could be used as evidence for vehicle j being either trustworthy or untrustworthy. In this example the number of elements is limited to three, trustworthy, untrustworthy and uncertain (either trustworthy or untrustworthy), where each element is associated with a belief mass value.

4.3 Detecting Change in Behavior

The combined trust score about vehicle j, i. e., TI_j, is continuously monitored in the cloud and a suitable technique is needed in module 3 to detect changes in TI_j. The most basic detection technique would be to trigger an alert when a certain threshold, for example TI_j below 0.4, is reached. Naturally, this approach does not provide the flexibility of finding more subtle changes in a vehicles' behavior. Observing the mean and raising an alert as soon as the mean deviates from a given threshold on the other hand would be very slow and may not detect slow but steady changes.

Aminikhanghahi and Cook [23] provide a survey of change detection techniques for time series data including machine learning algorithms. They categorize existing work in unsupervised and supervised methods that are further split in more detailed categories. For this framework we require an unsupervised method that does not require training with labeled data. Hence, we investigate the use of two unsupervised methods, namely CUSUM and one Bayesian detection approach.

Cumulative Sum (CUSUM) was originally proposed by Page [24] and has been adapted over the years to, for instance, support online detection [23,25]. This method is able to detect small and steady changes, as the cumulative sum of the deviations from a target value is calculated [25]. Granjon [26] describes the CUSUM algorithm, discusses practical considerations, such as choosing the detection threshold, and refers to other proposed variations of CUSUM.

Bayesian approaches for change point detection have, compared to CUSUM, the advantage of being faster; they require fewer samples for detecting change and have a lower computational cost [23]. In Section 5.3 we give more details and compare an implementation of CUSUM and Bayesian change detection.

4.4 Data Analysis in the Cloud

First, the data needs to be analyzed in order to design and apply appropriate anomaly detection techniques. Knowledge Discovery in Databases (KDD) [27] is a process for gathering new knowledge from data. This process consists of several steps starting from understanding the area of application to interpreting and evaluating patterns found in the data, thus gaining new knowledge. Data mining is one step in this process and supports the developer in identifying new patterns in the data by applying specific algorithms [27, 28].

Hemdan and Manjaiah [29] provide an overview of the principles of digital forensics, intrusion detection and suitable types of data science methods in the context of Internet of Things. The authors list prediction, classification, clustering and relation rule techniques as appropriate techniques for intrusion detection. Torres et al. [30] review machine learning techniques in the context of cyber security. They provide more detail and discussions on relevant techniques, such as self-organizing maps (SOM) and random forest. Kang [31] focuses on anomaly detection techniques for monitoring a product's health, however, the overview of machine learning techniques and discussions about the advantages and disadvantages of each category of machine learning techniques is still relevant for identifying techniques for module 4.

5 V2C Anomaly Detection Framework

We present a detailed description of each module and discuss the applicability of methods identified in Section 4 in the following subsections. Moreover, we propose how the selected methods can be adapted when necessary.

5.1 Trust Evaluation

In **module 1**, a trust score, often a value between 0 and 1, for each vehicle is computed every time a vehicle interacts or receives V2V messages, e. g., Cooperative Awareness Messages (CAM) [32]. The trust score needs to reflect the cooperativeness when performing cooperative actions, such as platooning, lane change and crossing an intersection. In addition, vehicles should also be evaluated based on the accuracy of the information they provide, e. g., speed, geographical position, driving intentions as well as correctness of warning messages. Such an evaluation of each vehicle's behavior and accuracy is typically performed using a trust or reputation model (see Section 4.1). The trust model should also be applied on the own vehicle's system, the ego vehicle.

Below, we discuss how trust models for calculating the trust scores of the ego vehicle and surrounding vehicles can be applied for the purpose of the *V2C Anomaly*

Detection framework. For instance, vehicle i should calculate its own trust score $TI_{i,i}$ and the trust score for vehicles it interacted with, i. e., $TI_{i,j}$. These trust scores are updated over a specified period and maintained in the vehicle's own database along with a confidence value indicating the level of confidence for each specific trust score. After the specified period, the computed trust scores and corresponding confidence values are uploaded to the cloud for further analysis.

Trust Model. The trust models in [17] and [16] both include an evaluation of the vehicle's own performance with respect to trust. Compared to Bißmeyer et al. [17], Rosenstatter and Englund [16] also provide a detailed discussion about the relevant behavior specific to the vehicle in front, surrounding vehicles it interacts with and the ego vehicle.

The trust score presented in [16] is considered by the ego vehicle for decision-making and comprises four trust scores with range $[0, 1]$ that are combined using a weighted average: (1) ego vehicle; (2) vehicle in front; (3) interacting vehicles; (4) environment. The authors differentiate between the vehicle in front and other vehicles it interacts with as the vehicle in front is most important from a safety perspective and since it can be easily evaluated with its own sensors like the front radar. Moreover, the authors have shown in real scenarios with sensor noises how such a trust score can represent the current situation respectively trust in the surrounding other vehicles.

In our *V2C Anomaly Detection* framework, the trust scores of the vehicles should be evaluated based on all available knowledge of the ego vehicle, similar to [16], and split into two distinct and one combined trust score:

$TI_{i,j}$	Trust score of vehicle j computed by vehicle i.
$TI_{i,i}/TI_{j,j}$	Own perceived trust score of vehicle i/j.
TI_j	Combined trust score of vehicle j (see Section 5.2).

Anomaly Identification. The selected trust model needs to be able to identify the attacks and anomaly types defined in the scenarios in Section 2.1. To further clarify situations where trust scores and the consequent analysis in the cloud are extremely useful, we will give one practical example for each scenario in Table H.2.

Update Frequency. The trust scores should ideally be calculated and updated upon receipt of new messages from relevant vehicles within the VANET. However, due to limited computational resources or the high number of vehicles identified as relevant, it might be necessary to reduce the computation of trust scores to a lower frequency. Relevant vehicles are defined as those in direct proximity of the ego vehicle, but also other vehicles it interacts with, such as vehicles farther away that send a warning message, vehicles participating in a platoon or vehicles cooperating to efficiently pass an intersection.

The trust score needs to be updated as the vehicles are periodically re-evaluated. This can be achieved by computing the moving average respectively the weighted moving average when new trust evaluations for a vehicle are computed in order to maintain a single trust score $TI_{i,j}$ per vehicle per evaluation period, e. g., per day.

Confidence $C_{i,j}$. Due to the fact that interactions with a specific vehicle can last from a few seconds to several hours when considering platooning, we propose to include a confidence value or counter ($C_{i,j}$) to each $TI_{i,j}$ to indicate the confidence in

Table H.2: Examples of how a trust model can identify attacks and anomalies.

Anomaly/Attack	Identification
Unauthorized Firmware Manipulation	
The attacker, such as the owner, upgrades the ECU firmware to let the automated vehicle (vehicle j) drive faster than the speed limit.	Surrounding vehicles are able to observe the speed of the affected vehicle using their own sensors, e. g., radar, and therefore reduce $TI_{i,j}$.
HW/SW failures	
The camera provides noisy images and therefore the vehicle's lane keeping assistant is misbehaving and causes the vehicle to bounce within the lane.	Surrounding vehicles observe the behavior (lateral movement within the lane) and therefore reduce $TI_{i,j}$. This behavior can be also detected by the vehicle itself.
Legitimate SW/HW update	
A legitimate update causes the vehicle to misinterpret the current road conditions, e. g., icy roads, and drives too fast.	Surrounding vehicles observe the higher speed and therefore reduce $TI_{i,j}$.
Cooperative Driving	
To disrupt traffic an attacker performs an unauthorized firmware modification so that the vehicle prevents other vehicles from performing a cooperative merge into one lane (i. e., does not allow vehicles to merge in front).	This behavior is mainly identified by vehicles that experience the denial of entering the lane in front of vehicle j, but also by other surrounding vehicles.

the calculated trust score per vehicle. This way, individual vehicles can also use this confidence value to decide whether they should keep (and store) the trust score and report it to the cloud in cases when the maximum capacity of the allocated memory for the trust scores is reached.

Kerrache et al. [20] include a factor that considers the number of verified legal/correct actions within their calculation of the direct trust score (corresponds to $TI_{i,j}$). We suggest to upload a confidence value, such as the total number of interactions, together with the trust scores in order to indicate the quality of the calculated trust score $TI_{i,j}$.

Role of $TI_{i,i}$. Continuously evaluating a vehicle's own reliability and performance is important as the own system acts as the reference (its own sensors and knowledge about the environment) when evaluating other vehicles. When a vehicle, for instance, experiences a sudden drop in its own trust score while interacting with other vehicles or driving autonomously it can automatically enable logging of the most recent events. Logs can be uploaded to the cloud along with the trust scores of the other vehicles it interacted with recently. Including evidence of such a situation is crucial for further investigations, when, for instance, other vehicles also report a lower trust score for this vehicle.

Role of $TI_{i,j}$. The assessment of the other vehicles can be split in two different categories: (i) verification of reported sensor information, e. g., position, speed, acceleration, lane; and (ii) behavior, e. g., correctly reporting warning messages, interaction/cooperation with other vehicles, speed according to laws and road conditions.

Evaluating not only the trust when interacting with other vehicles is important from a safety perspective as vehicles will rely on sensor information received from other vehicles during platooning and other cooperative scenarios.

Reporting Trust Scores. Each vehicle maintains its own database of trust scores. The database maintains a trust score per vehicle per defined period, e. g., per day, comprising of 4 columns: (1) vehicle ID, (2) date, (3) trust score ($TI_{i,i}$ or $TI_{i,j}$), and (4) confidence value ($C_{i,j}$).

This data is uploaded to the cloud periodically, e. g., on a daily basis. By including $C_{i,j}$, it is possible to further filter trust scores and consider whether this specific trust score should be considered in the aggregated trust score per vehicle or not.

Recommendation. Adapting the selected trust model (e. g., [16]) to detect the attack examples shown in Tables H.1 and H.2 with a significant change in the trust score is essential for this framework. In Table H.2 we showed how it is possible to identify such attacks. Moreover, we propose and discuss the types of trust scores, and how often they, including a confidence value, should be uploaded to the cloud.

5.2 Combining Trust Scores

Module 2 of the *V2C Anomaly Detection* framework is located in the cloud and receives trust scores ($TI_{i,j}$) from other vehicles about vehicle j periodically. Section 4.2 provides an overview of approaches to combine this knowledge, namely aggregation using mean and the use of Dempster's rule of combination.

Comparing the uses of arithmetic and geometric mean and considering the use case for this work as well as the range of the trust score, $[0, 1]$, we see the arithmetic mean to be more applicable since the expected data does not vary logarithmically.

The strength of DST and Dempster's rule of combination is to combine knowledge provided by different observers without the need of having complete knowledge of the prior probabilities. The methods discussed in Section 4.2 highlight when DST is applicable in the context of VANETs: *When modeling and combining the uncertainty about whether to trust or not trust a specific node or information received, e. g., a warning message.*

Use Case. Consider a case when vehicles report contradicting trust scores about vehicle j. These vehicles can either be honest and report a high trust score (0.9), or dishonest or incorrect and report a low trust score (0.2). Table H.3 shows the results with varying ratios of honest and dishonest vehicles. Comparing arithmetic and geometric mean shows that the geometric mean is more pessimistic and that a minority of dishonest or disagreeing vehicles have a higher impact on the combined trust score. As the *V2C Anomaly Detection* framework aims at identifying anomalies by observing the aggregated trust score reported by a majority of honest vehicles, it is desirable to use the arithmetic mean.

For the DST-based approach, we adapt Chen and Venkataramanan [21] and define three elements: Trust, Distrust, and Uncertainty. If a vehicle trusts vehicle j with

Table H.3: Comparison of the mean and DST with a varying ratio of honest ($TI_{i,j} = 0.9$) and dishonest ($TI_{i,j} = 0.2$) vehicles.

Reported Trust Scores $TI_{i,j}$		Arithmetic Mean	Geometric Mean	DST*	
0.9	0.2			$m(t)$	$m(d)$
100%	0%	0.90	0.90	1.00	0.00
80%	20%	0.76	0.67	1.00	0.00
60%	40%	0.62	0.49	1.00	0.00
50%	50%	0.55	0.42	0.97	0.03
40%	60%	0.48	0.37	0.39	0.61
20%	80%	0.34	0.27	0.00	1.00
0%	100%	0.20	0.20	0.00	1.00

*Dempster-Shafer theory: degree of belief; trust $m(t)$ and distrust $m(d)$

probability of α it results in the following basic belief masses: $m(t) = \alpha$; $m(d) = 0$; $m(u) = 1 - \alpha$; If a vehicle distrusts vehicle j it results in: $m(t) = 0$; $m(d) = \alpha$; $m(u) = 1 - \alpha$. In addition, we define that a $TI_{i,j}$ below 0.5 equals to distrust with a probability of $1 - TI_{i,j}$, e. g., $TI_{i,j} = 0.2$ in Table H.3 indicates a distrust of $m(d) = 0.8$ and uncertainty of $m(u) = 0.2$. The results when applying a DST approach similar to the one we described shows that DST does not provide a sufficient level of detail. For instance, when the belief is high in both cases, honest $m(t) = 0.9$ and dishonest $m(d) = 0.8$ vehicle, the uncertainty is still smaller than 10^{-2}.

Recommendation. Considering the behavior of these three different approaches we suggest applying the arithmetic mean for trust scores with a sufficiently high confidence value. For specific cases where the majority of vehicles report a low confidence value $C_{i,j}$, it can be tested whether a weighted average yields a better result when the change detection (module 3) is applied.

5.3 Detecting Change in Behavior

The observation of the historic development of the trust score is essential for this framework as an alert from **module 3** triggers further investigations. Section 5.1 defines the factors and means for computing a trust score reflecting other vehicles' and the own vehicle's behavior to identify anomalies in the scenarios described in Section 2.1. The combined trust score TI_j described in Section 5.2 is a one-dimensional time series with a new data point added each evaluation period, e. g., each day.

A variety of change point detection techniques are applicable for this kind of data (see Section 4.3). Figure H.3 shows a simulation of the aggregated trust score TI_j of vehicle j reported daily using the arithmetic mean over a period of one year with three different types of changes injected: (i) an immediate drop of 0.1 on day 28; (ii) an immediate increase of 0.2 on day 112; and (iii) a slow decrease of TI_j

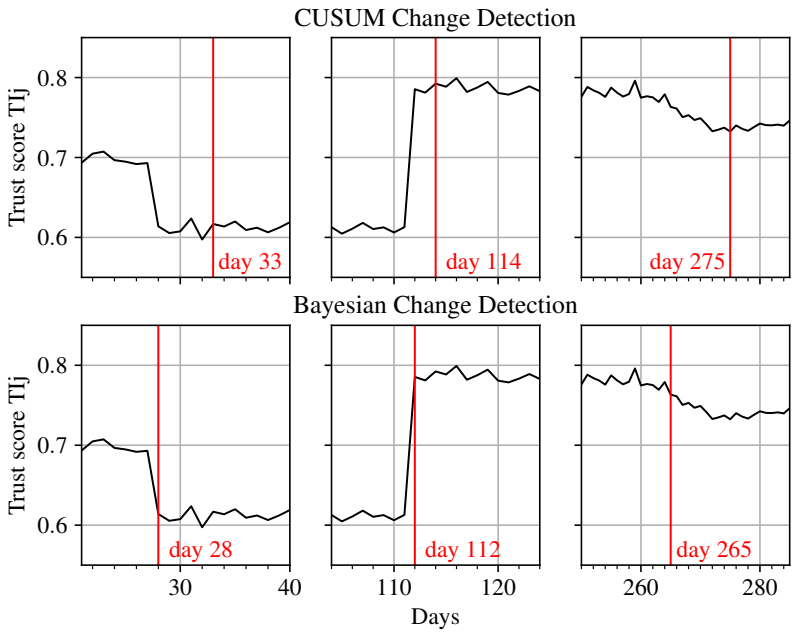


Figure H.3: Comparison of detection techniques applied to the trust score TI_j when 3 different types of changes occur.

where each day one more out of the 9 vehicles reports a decrease of the trust score of 0.05 starting from day 262.

We used the CUSUM detector implementation from Duarte [33] and modified it according to [25, p.40] describing the two-sided CUSUM algorithm. The Bayesian online change detection implementation from Kulick [34] following Adams and MacKay [35] was chosen as a second candidate.

Figure H.3 shows that both detection techniques were able to detect the changes of the trust score. For these specific injected changes CUSUM was 5 days respectively 2 days slower in detecting the changes (i) and (ii). Change (iii) is already detected by the Bayesian detection after 3 out of the 9 vehicles reported a $TI_{i,j}$ decreased by 0.05 whereas the CUSUM detected the change once TI_j stabilized.

Recommendation. The comparison of both techniques shows that the Bayesian change detection [34] outperforms the two-sided CUSUM detection [25, p.40]. This behavior is also confirmed by Aminikhanghahi and Cook [23], as they state that Bayesian approaches require less samples for detecting change and have a lower computational cost.

5.4 Data Analysis in the Cloud

An alert triggers **module 4** when a change in the trust score of individual vehicles in module 3 is detected. The trust scores ($TI_{i,j}$) reported by the affected vehicle will

be temporarily excluded in the computation of TI_j in module 2 to avoid including possibly wrong or forged trust scores. The cloud data is utilized by module 4 to investigate the cause of the anomaly and help to identify whether more vehicles in the fleet are affected. This cloud analysis can be split in two parts: (i) manual investigations; and (ii) automated anomaly & intrusion detection.

Manual Investigations. Initial manual investigations are necessary in order to identify and apply appropriate techniques for anomaly detection. Thus, it is important to follow a structured approach such as the Knowledge Discovery in Databases (KDD) process [27] which defines the steps for analyzing data. This process includes data mining as one step for applying statistics and machine learning techniques on the pre-processed data with the aim to discover new patterns and gain knowledge. As the type and the amount of data differs between the different entities, it is needed to follow such a process to being able to automate this analysis.

Anomaly and Intrusion Detection. Anomalies can be detected by deploying IDSs in the cloud. IDSs can be designed to detect misuse (knowledge-based [36]), i. e., signatures of known attacks, or to detect a change in behavior or specification (anomaly-based [36]).

A majority of anomalies and intrusions can be detected with specification-based techniques, such as specifications about the defined protocol handshakes, network protocol specifics and conformity to the application protocol. Other parameters that can be validated with such specifications are the location and the IP address from which the requests were sent and the exact time a workshop claimed to have performed an update.

Anomaly detection techniques based on artificial intelligence need to be adjusted for the dataset, thus it is important to follow a process such as KDD. In Section 4.4 we refer to publications that identify and categorize relevant machine learning techniques for anomaly detection. An example of a relevant solution is a method based on Isolation Forest [37] presented by Siddiqui et al. [38]. The introduced anomaly detection system produces not only an anomaly score, but also generates explanations for detected anomalies for the analysts and additionally includes a feedback loop to increase the detection performance. Another technique to consider are self-organizing maps (SOM). Qu et al. [39], for instance, explore SOM-based clustering techniques used for intrusion detection.

Recommendation. Considering the fact that vehicle manufacturers may have to monitor a fleet of more than a million vehicles and that many detection techniques have a high time complexity (see [28]), it may not be feasible to perform an on-line analysis of the entire data available in the cloud. Therefore, a scalable solution is required. The *V2C Anomaly Detection* framework reduces the computational costs by only triggering a search for anomalies in the cloud when an anomaly of the trust score, reported by several independent vehicles, is detected. Moreover, we suggest the following two steps:

1. Upon receiving an alert from module 3 the cloud data relevant to vehicle j needs to be analyzed in order to find more information about the anomaly and its cause.

2. Once patterns of anomaly or intrusion in the cloud data specific to the alert about vehicle j have been identified, the entire vehicle fleet or a subset, such as vehicles within a specific region, can be analyzed.

5.5 Vehicle Data in the Cloud

In this section we explore the types and variety of data available in the cloud. The type of data strongly depends on the actor performing the analysis and their terms of agreement with their customers. The actor is not limited to the vehicle manufacturer, but it may also be the owner of a vehicle fleet, e. g., logistics company, or an authority. We therefore identify and describe the type of data that should be considered for the analysis in module 4.

Data uploaded to the cloud can be split in four categories: (1) trust scores; (2) application data from services using the server back end; (3) diagnostic events, e. g., update events, negative response codes (NRCs) according to ISO 14229 [40]; (4) in-vehicle IDS alerts. Figure H.4 provides an overview and examples of these categories.

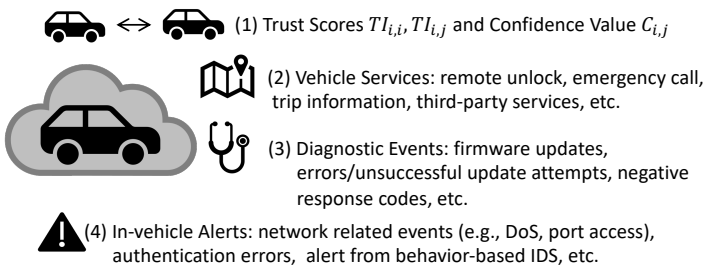


Figure H.4: Data accessible in the cloud

Vehicle Services. Many vehicle services, such as remote unlock, emergency call and smartphone application, require communication to the server back end. The data of these services can contain status updates from the vehicle, for instance, location, traffic and road conditions, and trip statistics. The information shared depends on the services the customer subscribes to as well as the user agreement for using private sensitive data for security analysis. Metadata from the communication between the vehicle and the server, and data from the application protocol and possible deviations from it can be additional evidence of an anomaly or intrusion.

Events and service information performed by an authorized workshop are also uploaded to the cloud in order to keep track of the modifications and repairs performed on the individual vehicles. Such data includes information about the workshop, time and location as well as which vehicle unit was upgraded, defect and replaced, and firmware versions of each vehicle unit.

In addition, third-party service providers may choose to share the identity or more information about anomalously behaving vehicles.

Diagnostic Events. Vehicles not only need to report that firmware updates have been performed, they also need to provide information about when and how, e. g., over-the-air or manually at the workshop, the new firmware was downloaded.

Furthermore, OEMs need to know whether the firmware update was successful and if it took more than one attempt to upgrade. This information can be used, for instance, to identify whether vehicles were subject to a possible intrusion attempt. ISO 14229 [40], *Road Vehicles – Unified Diagnostic Services (UDS)*, is widely used and also incorporated in the AUTOSAR system architecture.[1] From a security perspective UDS events are of high interest as these services are used to modify firmware and for short-term controls, e. g., triggering ECU restarts. Appendix A.1 of ISO 14229 [40, Part 1] lists the defined Negative Response Codes (NRCs). An example when such NRCs are sent is an event that the limit of failed authentication attempts has been reached or the attempt of sending specific UDS commands while the vehicle was driving faster than the specified limit.

In-vehicle IDS Alerts. In-vehicle IDSs are essential for detecting and subsequently preventing a large number of attacks. This need is also reflected in the *ENISA Good Practices for Security of Smart Cars* [9]. The detail of information when IDS alerts are received depends on the type of IDS. Thus, the information from alerts can range from detailed information about which attack has been recognized, to only an alert indicating that behavior outside the modeled normal behavior has been detected.

6 Discussion

To highlight the benefits of the *V2C Anomaly Detection* framework, we will discuss a use case similar to change (iii) in Section 5.3 to detail the tasks and discuss the challenges for each module. The introduced change causes the combined trust score TI_j to continuously drop as more vehicles report a lower trust score for vehicle j. Possible causes for this change could be either a malicious actor who performed an illegal modification of the firmware (scenario 1 in Section 2.1) or a legitimate firmware update (scenario 3).

(1) Trust Evaluation. Each day more vehicles observe an undesired behavior of the automated vehicle j, such as speeding or driving faster than what the current road condition allows. Therefore, the vehicles lower the individually perceived trust score $TI_{i,j}$ according to the trust model and report it together with their own trust score $TI_{i,i}$ and $C_{i,j}$ to the vehicle manufacturers' cloud.

Opening such a peer evaluation between vehicles to a group of vehicle manufacturers would naturally lead to more independent evaluations by other vehicles encountered on the roads and thus more data. However, the trust models need to be similar in terms of evaluation criteria in order to avoid biases of different implementations. Another challenge is the identification of vehicles when pseudonym certificates are used to hide the true identities of vehicles in the VANET. In this case the corresponding entity, i. e., certificate authority, needs to be involved in order to correlate the trust score to the correct vehicles.

(2) Combining Trust Scores. Module 2 combines the trust scores $TI_{i,j}$ with a sufficiently high confidence value $C_{i,j}$ in form of TI_j. In the beginning only a few

[1] `https://www.autosar.org`

vehicles may experience a change in the behavior as example 5 in scenario 3, for instance, is only perceived when the road is slippery or icy, e. g., in the mornings and evenings, however, over time more vehicles report a lowered $TI_{i,j}$ causing the overall trust score to drop more significantly. The trust score perceived by the vehicle itself, i. e., $TI_{j,j}$, may stay stable as the vehicle itself perceives the environment incorrectly due to the firmware update. For this reason, we suggest to monitor $TI_{j,j}$ separate from TI_j.

(3) Detecting Change in Behavior. With the TI_j degrading each day, module 3 detects the change and raises an alert once a certain threshold is reached. Sections 4.3 and 5.3 show that there are many suitable algorithms for detecting change in one-dimensional time series data. In this work we explored two techniques in more detail, namely CUSUM and Bayesian online detection. The comparison in Figure H.3 shows that the Bayesian online detection is faster compared to CUSUM as it detects the change already while the change is progressing. This characteristic is also described in the comparison of both techniques [23].

(4) Cloud Analysis. As soon as the observation of the combined trust score per vehicle TI_j triggers an alert raised by module 3, the anomaly detection defined in module 4 gets activated. In the first step, the anomaly detection performs an analysis of data related to vehicle j. In this context, a specification-based IDS may detect an unusually high number of unsuccessful firmware update attempts, possibly caused by an attacker, followed by a successful upgrade within a specified period. Another analysis can inspect whether the vehicle was recently in a workshop or if the firmware was upgraded. Based on these anomalies, the system can further search for vehicles with similar patterns in the database.

In addition to specification and rule-based intrusion detection mechanisms, machine learning techniques, such as classification and clustering techniques are appropriate candidates for this type of data. Nevertheless, it is necessary to follow an approach such as KDD to explore the data, find new patterns and consequently adapt existing detection techniques.

Evaluation. Representing undesired and malicious behavior in form of trust scores is essential for detecting anomalies using this framework. Existing trust models [16, 17] show how data plausibility checks and misbehavior detection can be used as a base for making decisions in automated vehicles. In Section 5.1, we describe the requirements for such trust models and specifics on how to use them including detailed examples highlighting what kinds of anomalies a trust model needs to be able to identify to significantly reduce the trust score. Moreover, we evaluated the modules on their own with initial experiments and discussions considering also the practicality of each module. A sophisticated prototype covering individual vehicles including the cloud data they produce is planned in future work.

7 Conclusion

In this paper, we present the *V2C Anomaly Detection* framework, which is a novel framework combining the assessment of Vehicle-to-Vehicle communication and the perceived quality of cooperative interactions between vehicles resulting in a

trust score, with vehicle data in the cloud. The peer evaluation of vehicle behavior allows identification of local anomalies and attacks even when important security controls such as in-vehicle IDSs fail to detect them, for example due to an attacker exploiting a vulnerability, an insider or a firmware upgrade causing unintended behavior. Furthermore, the analysis of cloud data makes it possible to detect and identify patterns of anomalies and intrusions on a wider scale such as on a fleet level. Ultimately, the advantage of the *V2C Anomaly Detection* framework lies in the fact that it is designed to reduce the computational costs in the cloud by triggering a cloud analysis once the combined trust evaluation performed by independent vehicles shows a significant change, i. e., a changed behavior resulting in a decline of the trust score.

We have provided scenarios focusing on persistent threats to explain the requirements for each module of the *V2C Anomaly Detection* framework in terms of functionality, inputs and outputs. We also provide an initial identification and detailed discussions to aid in choosing or adapting techniques for each module so that a vehicle manufacturer, fleet owner or other actor in the cloud is able to select and adapt relevant techniques depending on the available data.

Acknowledgements. This research was supported by the Vinnova-funded projects "CyReV (phase 1 & 2)" and "KIDSAM" and by the Swedish Civil Contingencies Agency (MSB) through the projects RICS2 and RIOT.

Bibliography

[1] A. Davila and M. Nombela, "Platooning - safe and eco-friendly mobility." *SAE Technical Paper 2012-01-0488*, 2012.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Security Symposium*. San Francisco, CA: USENIX, 2011, pp. 77–92.

[3] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, p. 91, 2015.

[4] M. Yan, J. Li, and G. Harpak, "Security Research Report on Mercedes-Benz Cars," *Black Hat USA*, p. 38, 2020. [Online]. Available: https://www.blackhat.com/us-20/briefings/schedule/index.html#security-research-on-mercedes-benz-from-hardware-to-car-control-20746

[5] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *Sixth International Conference on Information Assurance and Security*. Atlanta, GA: IEEE, 2010, pp. 92–98.

[6] N. Nowdehi, W. Aoudi, M. Almgren, and T. Olovsson, "CASAD: CAN-Aware Stealthy-Attack Detection for In-Vehicle Networks," 2019.

[7] S. Sanwald, L. Kaneti, M. Stöttinger, and M. Böhner, "Secure boot revisited: Challenges for secure implementations in the automotive domain," *SAE Int. J. Transp. Cyber. & Privacy*, vol. 2, no. 2, pp. 69–81, aug 2020.

[8] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "Darts: Deceiving autonomous cars with toxic signs," 2018.

[9] The European Union Agency for Network and Information Security (ENISA), "Good practices for security of smart cars," ENISA, Tech. Rep., Nov 2019.

[10] R. Hussain, J. Lee, and S. Zeadally, "Trust in vanet: A survey of current solutions and future research opportunities," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–19, 2020.

[11] P. Guo, H. Kim, L. Guan, M. Zhu, and P. Liu, "VCIDS: Collaborative intrusion detection of sensor and actuator attacks on connected vehicles," in *Security and*

Privacy in Communication Networks, X. Lin, A. Ghorbani, K. Ren, S. Zhu, and A. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 377–396.

[12] T. Nandy, R. M. Noor, M. Yamani Idna Bin Idris, and S. Bhattacharyya, "T-BCIDS: Trust-based collaborative intrusion detection system for VANET," in *2020 National Conference on Emerging Trends on Sustainable Technology and Engineering Applications (NCETSTEA)*, Durgapur, India, 2020, pp. 1–5.

[13] E. A. Shams, A. Rizaner, and A. H. Ulusoy, "Trust aware support vector machine intrusion detection and prevention system in vehicular ad hoc networks," *Computers & Security*, vol. 78, pp. 245–254, 2018.

[14] R. G. Engoulou, M. Bellaiche, T. Halabi, and S. Pierre, "A decentralized reputation management system for securing the internet of vehicles," in *International Conference on Computing, Networking and Communications (ICNC)*. Honolulu, HI: IEEE, 2019, pp. 900–904.

[15] S. A. Soleymani, A. H. Abdullah, M. Zareei, M. H. Anisi, C. Vargas-Rosales *et al.*, "A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing," *IEEE Access*, vol. 5, pp. 15 619–15 629, 2017.

[16] T. Rosenstatter and C. Englund, "Modelling the level of trust in a cooperative automated vehicle control system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1237–1247, 2018.

[17] N. Bißmeyer, S. Mauthofer, K. M. Bayarou, and F. Kargl, "Assessment of node trustworthiness in vanets using data plausibility checks with particle filters," in *Vehicular Networking Conference (VNC)*. Seoul, South Korea: IEEE, 2012, pp. 78–85.

[18] G. Shafer, "Dempster-shafer theory," *Encyclopedia of artificial intelligence*, vol. 1, pp. 330–331, 1992.

[19] T. Halabi and M. Zulkernine, "Trust-based cooperative game model for secure collaboration in the internet of vehicles," in *International Conference on Communications (ICC)*. Shanghai, China: IEEE, 2019, pp. 1–6.

[20] C. A. Kerrache, C. T. Calafate, N. Lagraa, J. Cano, and P. Manzoni, "Hierarchical adaptive trust establishment solution for vehicular networks," in *27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. Valencia, Spain: IEEE, 2016, pp. 1–6.

[21] T. M. Chen and V. Venkataramanan, "Dempster-shafer theory for intrusion detection in ad hoc networks," *IEEE Internet Computing*, vol. 9, no. 6, pp. 35–41, 2005.

[22] C. Zhang, K. Chen, X. Zeng, and X. Xue, "Misbehavior detection based on support vector machine and dempster-shafer theory of evidence in vanets," *IEEE Access*, vol. 6, pp. 59 860–59 870, 2018.

[23] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 339–367, 2017.

[24] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[25] M. Basseville and I. V. Nikiforov, *Detection of abrupt changes: theory and application*. Englewood Cliffs, NJ: Prentice Hall, 1993, vol. 104.

[26] P. Granjon, "The cusum algorithm-a small review," GIPSA-lab, Tech. Rep. hal-00914697, 2013.

[27] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, no. 3, p. 37, Mar. 1996. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/1230

[28] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[29] E. E. Hemdan and D. H. Manjaiah, *Cybercrimes Investigation and Intrusion Detection in Internet of Things Based on Data Science Methods*. Cham: Springer International Publishing, 2018, pp. 39–62. [Online]. Available: https://doi.org/10.1007/978-3-319-70688-7_2

[30] J. Martínez Torres, C. Iglesias Comesaña, and P. J. García-Nieto, "Review: machine learning techniques applied to cybersecurity," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 10, pp. 2823–2836, Oct 2019. [Online]. Available: https://doi.org/10.1007/s13042-018-00906-1

[31] M. Kang, *Machine Learning: Anomaly Detection*. Hoboken, NJ: Wiley-IEEE Press, 2019, ch. 6, pp. 131–162.

[32] ETSI, "Basic set of applications; part 2: Specification of cooperative awareness basic service," ETSI, Intelligent Transport Systems (ITS) – Vehicular Communications EN 302 637-2 V1.3.2, 2014.

[33] M. Duarte, "detecta: A python module to detect events in data," https://github.com/demotu/detecta, 2020, visited on 2020-11-12.

[34] J. Kulick, "Bayesian changepoint detection – python implementation," 2020, visited on 2020-11-12. [Online]. Available: https://github.com/hildensia/bayesian_changepoint_detection

[35] R. P. Adams and D. J. C. MacKay, "Bayesian online changepoint detection," 2007.

[36] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805 – 822, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128698000176

[37] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Eighth International Conference on Data Mining*. Pisa, Italy: IEEE, 2008, pp. 413–422.

[38] M. A. Siddiqui, J. W. Stokes, C. Seifert, E. Argyle, R. McCann *et al.*, "Detecting cyber attacks using anomaly detection with explanations and expert feedback," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE, 2019, pp. 2872–2876.

[39] X. Qu, L. Yang, K. Guo, L. Ma, M. Sun, M. Ke, and M. Li, "A survey on the development of self-organizing maps for unsupervised intrusion detection," *Mobile Networks and Applications*, Oct 2019. [Online]. Available: https://doi.org/10.1007/s11036-019-01353-0

[40] I. O. for Standardization (ISO), "ISO 14229:2020 Road vehicles – Unified diagnostic services (UDS)," ISO, Standard, 2020.